Doc. No. 130071648

M-970

DDP-516-11
HIGH-SPEED
ARITHMETIC UNIT
Option Manual

September 1966

Honeywell

COMPUTER CONTROL
DIVISION

Original printing, Sept. 1966
Reprinted April 1967, July 1967, Nov. 1967

Printed in U.S.A.

Published by the Publications Department,
Honeywell Inc., Computer Control Division

CONTENTS

Page

Introduction 1

Reference Data 1

Physical Characteristics 1

Functional Description 1

Installation 2

Theory of Operation 2

General Description 2

Detailed Description 2

Multiply 2

Divide 8

Normalize 10

Shift Count to A 11

Enter Double-Precision Mode 12

Enter Single-Precision Mode 12

Double Load 12

Double Store 13

Double Add 13

Double Subtract 15

List of Parts for DDP-516-11 High Speed Arithmetic Option 17

Appendix A Flow Charts/Instruction Analyses A-1

ILLUSTRATIONS

Page

1 High-Speed Arithmetic Unit PAC Locations in the DDP-516 Central Processor Unit 3

2 Double Shifting the A- and B-Registers 6

3 Shifting the A- and B-Registers for B16 = 17 7

4 Forming the Product MADFF Set 7

5 Forming the Product MADFF Reset 7

6 Divide Termination 11

7 Double Add, Simplified Diagram 14

8 Double Subtract, Simplified Diagram 16

DDP-516-11
HIGH-SPEED ARITHMETIC UNIT
OPTION

INTRODUCTION

This document provides a technical description of the High-Speed Arithmetic Unit Option for the DDP-516 General Purpose Computer. The option enhances the arithmetic capability of the central processor unit (CPU) by providing hardware implementation of multiply, divide, and normalize. It also provides double-precision load, store, add, and subtract functions. A total of 10 instructions are involved in the use of this option.

Reference Data

Title	Doc. No.
Instruction Manual for the DDP-516 General Purpose Computer, Volume I, Section II	130071620
Instruction Manual for the DDP-516 General Purpose Computer, Volume II	130071621
Instruction Manual for the DDP-516 General Purpose Computer, Volume III	130071622
Installation Manual for the DDP-516 General Purpose Computer	130071625
Programmers Reference Manual for the DDP-516 General Purpose Computer	130071585

Physical Characteristics

The High-Speed Arithmetic Unit Option consists of μ-PACs located in the CPU tilt-out assembly (A1). All interface wiring between the option and the main frame is point-to-point. No connectors are used.

Functional Description

The High-Speed Arithmetic Unit Option adds 10 instructions to the DDP-516 instruction repertoire. They are:

Multiply (MPY)
Divide (DIV)
Normalize (NRM)
Shift Count to A (SCA)
Enter Double Precision Mode (DBL)

Enter Single Precision Mode (SGL)
Double Load (DLD)
Double Store (DST)
Double Add (DAD)
Double Subtract (DSB)

All double-precision data are represented by two adjacent words of 16 bits each. The first word contains the sign and most significant half of the data; the second word has a ZERO sign bit, followed by the least significant half of the data. The first word is held in the main frame A-register. The second word is held in the main frame B-register. In memory, the first word is stored in an even location and the second word is stored in the next higher numbered odd location.

Instructions which reference double-precision operands must produce even, effective, addresses (after all indirection and indexing). An odd effective address will cause the instruction to be executed as if it had the next lower even effective address in the case of double load, add or subtract. An odd effective address in a double-precision store will cause the B-register content to be stored in the specified location without affecting any other register location.

INSTALLATION

All interconnections are hand wired. PAC locations for the High-Speed Arithmetic Unit Option are shown on Figure 1 (Dwg. No. 3016173). PAC descriptions are found in the Instruction Manual, Volume 1, Appendix A, Computer Control Division Doc. No. 130071620.

THEORY OF OPERATION

General Description

The theory of operation for the High-Speed Arithmetic Unit Option consists of a discussion for each of the 10 instructions and a flow chart and instruction analysis for each instruction. Logic drawings referenced in the analysis can be found in the Instruction Manual for the DDP-516 General Purpose Computer, Volume III, Computer Control Division Doc. No. 130071622. Reference should be made to the function index in the Instruction Manual for the DDP-516 General Purpose Computer, Volume II, Computer Control Division Doc. No. 130071621.

Detailed Description

Multiply (MPY)

In the multiply instruction, the contents of the A-register are the multiplier for a word stored in the memory. At the conclusion of the multiplication, the product is stored in the A- and B-registers; the sign and 15 most significant bits (MSB) are stored in the A-register, and the 15 least significant bits (LSB) are stored in the B-register. Bit 1 of the B-register is made a ZERO by convention.

In applying the rules governing the multiplication algorithm, the process starts at the low-order end of the multiplier. Shifting is to the right. If the LSB is a ONE, it is treated as though it had been approached by shifting across ZEROs.

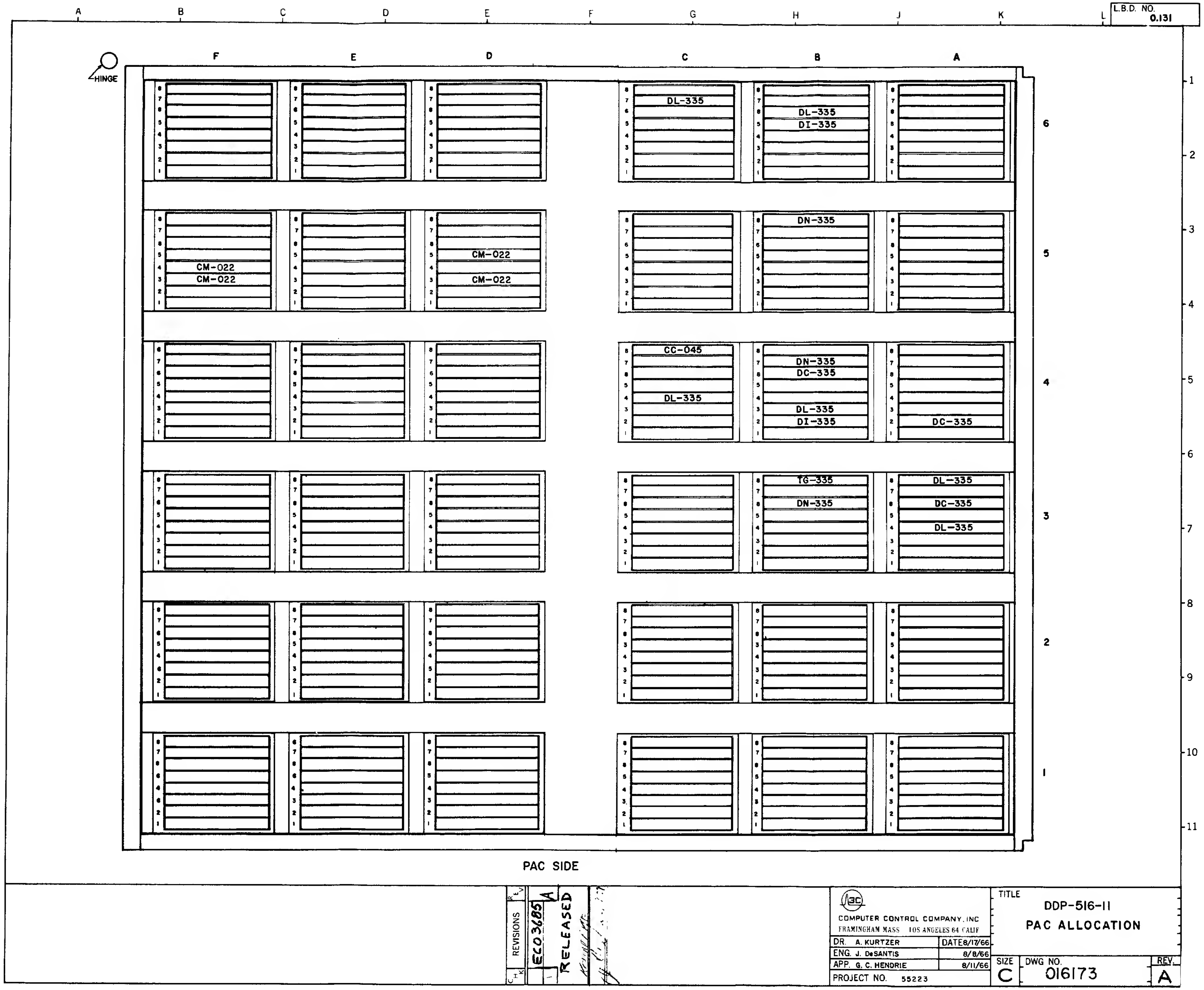


Figure 1. High-Speed Arithmetic Unit PAC Locations in the DDP-516 Central Processor Unit

Rule 1. -- When shifting across ZEROs, stop at the first ONE, and if the ONE is followed immediately by a ZERO, add the multiplicand and shift across all following ZEROs. If the ONE is followed immediately by a second ONE, subtract the multiplicand and shift across all following ONEs.

If the LSB is a ZERO, it is treated as though it had been approached by shifting across ONEs.

Rule 2. -- When shifting across ONEs, stop at the first ZERO and if the ZERO is followed immediately by a ONE, subtract the multiplicand and shift across all following ONEs. If the ZERO is followed immediately by a second ZERO, add the multiplicand and shift across all following ZEROs.

The foregoing operations are implemented in the computer as follows:

The algorithm is modified to permit the processing of two multiplier bits per shift cycle (duration of one shift cycle = 0. 48 μsec). Each shift cycle starts at time T3 and ends at T2, except for the last shift cycle which begins at T3 and ends at T4. There are eight such cycles per multiply instruction, seven from T3 to T2 and one from T3 to T4.

The arithmetic operation does not begin until T3 of the first pass through the A-cycle. (Refer to the multiply flow chart.) T1 and T2 are used for initialization. At T3 note that B16 and B17 are tested for equality. (B17 is the A00FF.) Since this is the first pass, B17 is known to be a ZERO (A00FF is cleared by CLATR- at T2; see instruction analysis for multiply.) B16 can be in either state. If B16 and B17 are unequal, the MADFF is reset and B15 is tested. For this discussion it is assumed that B16 and B17 are unequal.

B15 is tested and assumed to be a ONE in this case. Following rule 1 of the multiply algorithm, the multiplicand is subtracted from (A), and the remainder stored in the D-register.

NOTE

> While the previous operation is a subtractive process, all two's complementing arithmetic operations are additions. The subtraction occurs when the contents of the M-register are complemented. Note also that the M-register contains the multiplicand. The transfer from the [EA] occurred at T2.

This is correct since rule 1 states that when shifting across ZEROs, stop at the first ONE (B16 in this case). If the ONE is followed immediately by a ONE, subtract the multiplicand. When T2 is repeated due to the non-zero content of the shift register, the second part of the rule is implemented. During the repeat of T2, the state of the MADFF is tested. Since it was reset earlier, the exit path is through NO. This leads to the double shifting of the A- and B-registers by way of the D- and E-registers. Refer to Figure 2 for an illustration of this operation. Following this operation, the shift counter is incremented and T3 is re-entered.

The next example describes the multiply operation when the MADFF is set at T3. To set the MADFF, B16 and B17 must be equal (see flow chart). Note that all three possible paths out of the setting of the MADFF and the testing of B15, B16, and B17 include, as one of their functions, signal SRSTL+. This signal implements an arithmetic shift of the A-register into the adder as opposed to an arithmetic shift of the adder output to the

A-register as implemented by SRATS+ (see LBD No. 101 through 116). The object at this point is to double (M) and this is executed by halving (A).

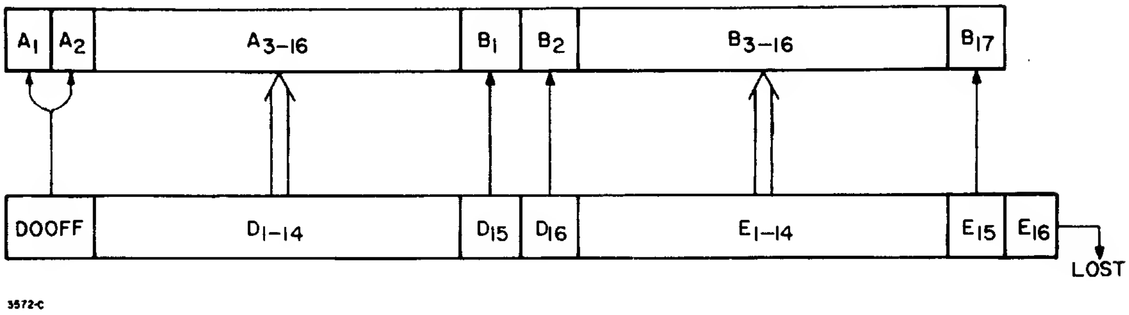


Figure 2. Double Shifting the A- and B-Registers

For discussion purposes, the states of B15, B16, and B17 are chosen as ZERO, ONE, ONE, respectively. This leads to the set of conditions (at T3) which implement the following general equation:

$$\frac{(A) + k(M)}{4}, \text{ where k can be any integer in the range } \pm 2.$$

NOTE

> This equation applies to all five cases in T3. It is introduced at this time to give the reader another approach to analysis of the MPY instruction.

In the example chosen for this discussion, k = +2, which reduces the equation to $\frac{1/2\ (A) + (M)}{2}$ (equation 1). The numerator of this equation is stored in the D-register as is shown on the YES exit path for the test of $B16 \cdot B17 = 1$?

Following the above operation, the shift counter is tested and found to be non-zero, causing T2 to be repeated. This leads to the test of the state of the MADFF which is known to be set at this time. This causes the single shifting (SRATS) of the A-register (which supplies the denominator of equation 1), and the double shifting of the B-register via the D- and E-registers. Refer to Figure 3 for an illustration of this operation.

Decisions similar to those just described are made repeatedly as the MSB of the multiplier are shifted down into lower bit positions and are examined as bits B15, B16, and B17. The process is terminated when the content of the shift counter is ZERO (see the end of T3 on the MPY Flow Chart). With SC = 0, T4 is entered to complete the last shift cycle.

At T4 the product is formed and stored in the A- and B-registers. The MADFF is tested for its state and the appropriate exit path is taken. Figures 4 and 5 illustrate these operations in each case.

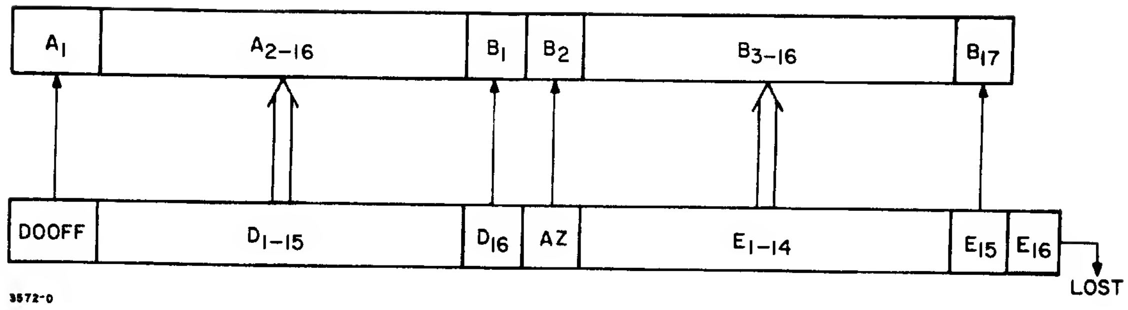


Figure 3. Shifting the A- and B-Registers for B16 = B17

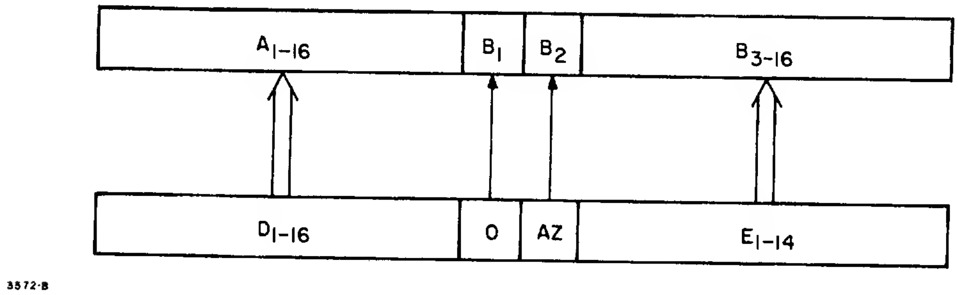


Figure 4. Forming the Product, MADFF Set

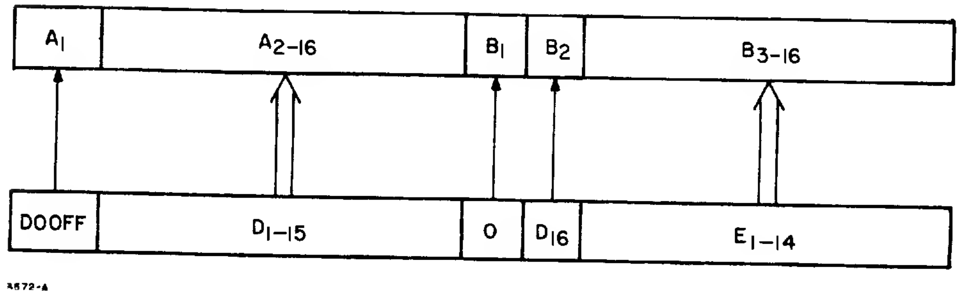


Figure 5. Forming the Product, MADFF Reset

Divide (DIV)

In the divide instruction, the sign and the most significant half of the dividend is contained in the A-register, bits 1 through 16. The least significant half of the dividend is contained in the B-register, bits 2 through 16. (Bit 1 of the B-register is ignored.) The divisor is a word stored in memory. The 16-bit quotient replaces the contents of the A-register, bits 1 through 16. The remainder (either zero or with the same sign as the dividend) replaces the contents of the B-register, bits 1 through 16.

If the initial magnitude of the A-register is equal to or greater than the magnitude of the effective operand, the overflow bit (CB1TF) is set and the computer proceeds to the next sequential instruction.

The divide instruction, unlike the multiply instruction, processes one quotient bit per shift cycle. (Refer to the multiply discussion for the definition of a shift cycle.) The instruction consists of an F-cycle and an extended A-cycle. The F-cycle sets up the initial conditions for the divide operation. The initialization consists of resetting the A00FF, MADFF, and D0GFF, setting the CB1TF, and jamming the shift counter to octal 57. (Refer to Appendix A for the divide flow chart and instruction analysis.)

Entry into the A-cycle causes the sign of the dividend to be stored in the AZZZZ flip-flop and A00FF at T1.

NOTE

> Simple operations such as clearing registers, etc., are not described in this discussion. This is done to highlight significant operations as a supplement and analysis rather than cloud the discussion with operations apparent to the reader.

At T2 the divisor is fetched from memory and stored in the M-register. During T2, the shift counter is incremented from octal 57 to octal 60. (Keep the octal 60 in mind for a subsequent test of the contents of the shift counter.)

At TLATE (the OR of T2 and T3) the state of the MADFF is tested. Since it was reset as part of the F-cycle initialization and remains so until some time later in the instruction, the exit path must be to a test of A00FF = M01FF?. The function of this test is to determine whether the remainder and divisor have like signs or not. Since this is the first pass and there is no "remainder", the sign of the dividend is compared with that of the divisor; the dividend is the effective "remainder" at this time.

Either of two conditions satisfy the YES exit; one condition is satisfied when both the dividend and divisor are + and the other is satisfied when both the dividend and divisor are -. The indicated operation is a subtraction. This defines the first half of rule 1. (See Table 1 for Basic Rules of Division.) The other half of the rule states if the dividend and divisor are of unlike sign, add one to the other. The object of this rule is to combine the dividend and divisor in such a manner as to approach a remainder of zero.

Of significance in T3 is the fact that T2 is going to be repeated due to the reset state of the D0GFF. (Both the MADFF and D0GFF remain reset until the shift counter advances to at least octal 77 for MADFF and octal 00 for D0GFF.)

Table 1.
Basic Rules for Division

Rule	Definition
1. a.	If the remainder and divisor are equal in sign (both plus or both minus) during TLATE, subtract the divisor from the remainder, and generate a ONE quotient bit.
b.	If the remainder and divisor are unequal in sign during TLATE, add the divisor to the remainder, and generate a ZERO quotient bit.
2.	If the sign of the remainder is equal to that of the dividend when (SC) = 60_8, terminate manipulation of dividend and indicate improper divide (CB1TF set).
3.	During a proper divide, shift the A- and B-registers left for each case of (SC) = 60_8 through 76_8 and repeat rule 1.
4. a. 1	Divide termination (A = dividend, D = divisor) +A/+D ---- If the remainder is zero or positive, the quotient and remainder are correct as they stand and the division is complete.
a. 2.	If the remainder is negative, the divisor must be added to it. The quotient and remainder are then correct.
b. 1.	-A/+D ---- If the remainder is zero, the quotient and remainder are correct and the division is complete.
b. 2.	If the remainder is negative, it may or may not be correct. For a complete test, the divisor must be added to it. If the resulting value of the remainder is zero, the remainder and quotient are correct and the division is complete. If, however, the resulting value of the remainder is positive, the original value was correct and a subtraction must be performed to extract the original remainder.
b. 3.	If the remainder is positive, the divisor must be subtracted from the remainder, giving a negative remainder to complete the division.
c. 1.	+A/-D ---- If the remainder is zero or positive, it is correct as it stands.
c. 2.	If the remainder is negative, the divisor must be subtracted from it to make it correct.
d. 1.	-A/-D ---- If the remainder is zero it is correct.
d. 2.	If the remainder is positive, the divisor must be added to it to make it· correct.
d. 3.	If the remainder is negative, it may or may not be correct. To complete the test, the divisor must be subtracted from it. If the resulting value of the remainder is zero, it is correct. If the resulting value is positive, the original value is correct and the divisor must be added to recover the original value.
5.	If the quotient differs in sign from the divisor at T4, the quotient must be incremented by one.

Special notice should be taken of the arrangement of the exits from the test SC = ? when T2 is repeated. Note that the exits are arranged in ascending order, reading from left to right, to correspond to the incremented contents of the shift counter. Further, only one exit can be achieved at any given time. The only exit path possible at this time is octal 60.

This exit leads to a test of the signs of the dividend and remainder. The object is to determine the magnitude of the divisor as compared to the dividend. If, as a result of the addition or subtraction during TLATE, the remainder has not changed sign (D1QAZ), an improper divide is in progress. If the divide were allowed to continue, the dividend would be destroyed. In order to leave the dividend intact, the D1QAZ YES exit is taken to invalidate the instruction by looping through the remaining shift cycles while not operating on the dividend. Note that the CB1TF remains set, indicating an improper divide. An exception to the above exists where the dividend is lost. (This special case exists when the quotient is 077777 before rule 5 is applied. When the quotient is incremented by one (rule 5) an overflow occurs, setting the CB1TF to indicate an improper divide.

If D1QAZ is NO, the CB1TF is reset, the A- and B-registers are shifted left, and the shift counter is incremented. As part of the left shift action, a bit of the quotient is formed and injected into B16 ($D_1 \oplus M_1 \rightarrow B_{16}$). When TLATE and T3 are re-entered, decisions similar to those previously described are made to continue the division. Since this is a repetitive operation, its occurrence is assumed in the remainder of the discussion. Further, it is assumed that a proper divide is in progress. This means that no further mention is made of the CB1TF = 0 test.

With the above proviso in mind, the events during (SC) = 61_8 through 76_8 merely shift the A- and B-registers left, forming successive quotient bits, followed by an addition or subtraction, as indicated by rule 1.

Assume that the contents of the shift register are now equal to octal 77. This causes the sign of the remainder to be stored in the A00FF and the B-register to be shifted left. The B-register, but not the A-register, is shifted to fill the previously ignored bit 1 position of the B-register. Next, the remainder is examined (REM0K). Either of two conditions leads to the setting of the MADFF. They are (D) = 0, or $(D)_1$ = (AZ) = 0. Figure 5 illustrates the possible combinations of remainder and dividend and the action required, if any, to terminate in a proper divide.

With MADFF set and D0GFF still known to be reset, (A) is transferred to the D-register without manipulation, and the (SC) is tested and found to be octal 00. This is the divide terminate phase of the instruction. (See rule 4 in Table 1, and Figure 6.)

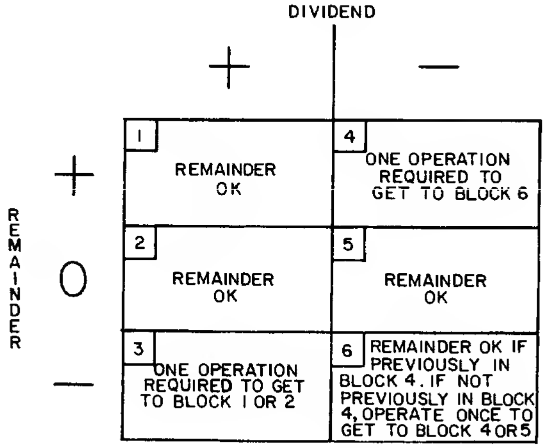
When the remainder is OK, the quotient and remainder are interchanged (see DIV flow chart) and the D0GFF is set. This sets up the conditions for MADFF YES and D0GFF = 1 in TLATE. Next, the quotient and divisor are checked for like signs. If the quotient differs in sign from the divisor, the quotient is incremented by 1 (rule 5). If the signs are the same, a simple transfer takes place.

At T4, the test D00 = D01? is made to test for a special case, described earlier in the discussion, wherein the quotient at the last TLATE was 077777.

Normalize (NRM)

The Normalize (NRM) instruction is used to change a floating point result so that the exponent and the mantissa lie in the standard normal range. This instruction considers the

A-register and the 15 magnitude bits of the B-register to be one 31-bit register. (Bit 1 of the B-register is ignored.) The A-register contains the most significant half of the number and the sign. The B-register contains the least significant half of the number. Bits 2 through 16 of both registers are shifted left until bits A01 and A02 are not equal.


DIVIDEND
+
—
REMAINDER
+
0
—
1
REMAINDER OK
4
ONE OPERATION REQUIRED TO GET TO BLOCK 6
2
REMAINDER OK
5
REMAINDER OK
3
ONE OPERATION REQUIRED TO GET TO BLOCK 1 OR 2
6
REMAINDER OK IF PREVIOUSLY IN BLOCK 4. IF NOT PREVIOUSLY IN BLOCK 4, OPERATE ONCE TO GET TO BLOCK 4 OR 5

NOTE:
DIVIDEND DOES NOT CHANGE SIGN DURING DIVIDE, BUT REMAINDER DOES.

8371

Figure 6. Divide Termination

If the number is ZERO, 32 shifts are performed before the instruction is terminated. (This represents an elapsed time of approximately 16.32 μsec.) Bits shifted out of bit position 2 of the B-register enter bit position 16 of the A-register. ZEROs are shifted into bit position 16 of the B-register. The sign of the number is retained throughout the instruction. The number of positions shifted is stored in the E-register. The contents of the E-register are made available with the SCA instruction (Shift Count To A).

Refer to the Normalize flow chart and instruction analysis for a detailed description of this instruction. Note that shifting D02 into A01 does not change the value of A01, since the shift occurs only when prior testing has found A01 = A02.

Shift Count To A (SCA)

The Shift Count To A (SCA) instruction places the contents of the E-register into the A-register. This involves only bits 11 through 16 of these registers. (Recall that the number of shifts performed in the normalize (NRM) instruction was stored in E-register at TL4.)

The reason for storing the number of shifts in the E-register is that an F-cycle follows the NRM instruction and in so doing, the contents of the shift counter are lost as a function of clearing the shift counter. The shift counter is always cleared at TL1 of every F-cycle. This loss of information is circumvented by placing the number of shifts in the E-register (during NRM) for subsequent transfer to the A-register during the SCA instruction. This means that, if the number of shifts is required for subsequent instructions, an SCA instruction should follow the NRM before the E-register's contents are destroyed by an IAB, MPY, DIV, or any shift or double-precision instruction.

Refer to the SCA flow chart and instruction analysis for a detailed description of this instruction.

Enter Double-Precision Mode (DBL)

This instruction causes all subsequent LDA, STA, ADD and SUB instructions to be executed in double-precision mode. This condition persists until an SGL instruction is executed or until the MSTR CLEAR button on the console front panel is depressed.

The instruction is a straightforward generic instruction with the DPM0D flip-flop (double-precision mode) set at TL3. Bit 13 on the console display is illuminated to denote operation in a double-precision mode. (The OP button must be depressed.) Refer to the DBL flow chart and instruction analysis for a detailed description of this instruction.

Enter Single-Precision Mode (SGL)

This instruction causes all subsequent LDA, STA, ADD, and SUB instructions to be executed in single-precision mode (normal operation). The effect of any prior DBL instruction is cancelled by resetting the DPM0D flip-flop and extinguishing bit 13 on the console display (OP button depressed). Refer to the Enter Single-Precision Mode flow chart and instruction analysis for a detailed description of this instruction.

Double Load (DLD)

The double load (DLD) instruction is identified by the same Op code as that of the load A (LDA) instruction. One or the other of these instructions is executed depending on whether the CPU is currently in the double-precision or single-precision mode of operation. (Refer to the DBL and SGL instructions described earlier in this manual.)

The DLD instruction requires three cycles for execution. They are an F-cycle and two A-cycles. (Refer to DLD flow chart.) Entry into the second A-cycle is a function of the contents of the shift counter.

The B-register is loaded first. This is done by loading the A-register with the ([EA + 1]) during the first A-cycle and transferring the contents of the A-register into the B-register via the adder and D-register. The EA is restored by clearing the least significant bit of the Y-register, a function of signal E0Y16-. The [EA] is loaded into the A-register during the latter part of the second A-cycle.

The method of decrementing (Y) used to restore the EA during this instruction demands that the EA + 1 be an odd-numbered location and the EA be an even-numbered location.

Some "don't care" operations are performed during T1 and T2 of the first A-cycle and should be ignored. These operations are the transfer of the contents of the A-register to the B-register via the adder and D-register. These operations are only necessary for the second A-cycle (see flow chart).

Double Store (DST)

The double store (DST) instruction is identified by the same Op Code as that of the store A (STA) instruction. One or the other of these instructions is executed depending on whether the CPU is currently in the double-precision, or single-precision mode of operation. (Refer to the DBL and SGL instructions described earlier in this manual.)

The DST instruction requires three cycles for execution. They are an F-cycle and two A-cycles. (Refer to DST flow chart.) Entry into the second A-cycle is a function of the contents of the shift counter.

The [EA] is accessed first, and is then loaded with the contents of the A-register. This occurs in the first A-cycle. Later in this same A-cycle, the contents of the A- and B-registers are interchanged. This is done because there is no path from the B-register to the memory other than through the A-register.

During the latter part of the first A-cycle, the least significant bit of the Y-register is set to ONE, a function of signal Y16FF-. This action enables access to the [EA + 1].

The contents of the A-register (initially the contents of the B-register due to the interchange of contents during the first A-cycle) is stored into the [EA + 1] during the second A-cycle. Later in the second A-cycle, the contents of the A- and B-registers are once again interchanged to restore the original contents of these registers.

Double Add (DAD)

The double add (DAD) instruction is identified by the same Op Code as that of the add (ADD) instruction. One or the other of these instructions is executed depending on whether the CPU is currently in the double-precision or single-precision mode of operation. (Refer to the Enter Double-Precision Mode (DBL) and Enter Single-Precision Mode (SGL) instructions described earlier in this manual.)

The DAD instruction requires three cycles for execution. They are an F-cycle and two A-cycles. (Refer to DAD flow chart.) Entry into the second A-cycle is a function of the contents of the shift counter.

Refer to the DAD flow chart and Figure 6 and note that the addition consists of two separate operations. Note also that it is always the contents of the A-register which is added to the double-precision word from memory.

If the low-order sum includes a carry out (B01 is set), the carry out is included in the addition. (See step 4, Figure 7.) In step 5, B01 is cleared since, by definition, B01 is always ZERO in a double-precision word.

1. LET: $(A) = a$, and

$(B) = b$,

$(A) \rightleftarrows (B)$

$\therefore$ $(A) = b$, and

$(B) = a$.

2.

$$\begin{array}{ll} & \qquad b \\ + & [EA + 1] \\ \hline \multicolumn{2}{l}{\}b + [EA + 1]\{} \end{array} \rightarrow (A) \qquad \text{(LOW ORDER SUM)}$$

3. $(A) \rightleftarrows (B)$

$\therefore$ $(A) = a$, and

$(B) = \}b + [EA + 1]\{$

4. $B01 = 0?$

YES:

$$\begin{array}{ll} & \qquad a \\ + & [EA] \\ \hline \multicolumn{2}{l}{\}a + [EA]\{} \end{array} \rightarrow (A) \qquad \text{(HIGH ORDER SUM)}$$

NO:

$$\begin{array}{ll} & \qquad a \\ + & [EA] + 1' \\ \hline \multicolumn{2}{l}{\}a + 1' + [EA]\{} \end{array} \rightarrow (A) \qquad \text{(HIGH ORDER SUM)}$$

WHERE: $1' = E1K17- = B01$

5. $0 \rightarrow B01$

3608

Figure 7. Double Add, Simplified Diagram

Double Subtract (DSB)

The double subtract (DSB) instruction is identified by the same Op Code as that of the subtract (SUB) instruction. One or the other of these instructions is executed depending on whether the CPU is currently in the double-precision or single-precision mode of operation. (Refer to the DBL and SGL instructions described earlier in this manual.)

The DSB instruction requires three cycles for execution. They are an F-cycle and two A-cycles. (Refer to DSB flow chart.) Entry into the second A-cycle is a function of the contents of the shift counter.

Refer to the DSB flow chart and Figure 8, and note that the subtraction consists of two separate additions. Note also that it is always the contents of the A-register which is added to the complemented double-precision word from memory.

If the low-order difference includes a carry out (B01 is reset), the carry out is included in the addition. (See step 4, Figure 8.) In step 5, B01 is cleared since, by definition, B01 is always ZERO in a double-precision word.

1. LET: $(A) = a$, and

$(B) = b$,

$(A) \rightleftarrows (B)$

$\therefore$ $(A) = b$, and

$(B) = a$.

2.
$$\begin{array}{r} b \\ + \qquad [\overline{EA+1}] + 1' \\ \hline \}b + 1' + [\overline{EA+1}]\{ \end{array} \rightarrow (A)$$

WHERE: $1' = E1K17-$

(LOW ORDER DIFFERENCE)

3. $(A) \rightleftarrows (B)$

$\therefore$ $(A) = a$, and

$(B) = \}b + 1' + [\overline{EA+1}]\{$

4. B01 = 0?

YES:
$$\begin{array}{r} a \\ + \qquad [\overline{EA}] + 1'' \\ \hline \}a + 1'' + [\overline{EA}]\{ \end{array} \rightarrow (A)$$

WHERE: $1'' = E1K17- = \overline{B01}$

(HIGH ORDER DIFFERENCE)

NO:
$$\begin{array}{r} a \\ + \qquad [\overline{EA}] \\ \hline \}a + [\overline{EA}]\{ \end{array} \rightarrow (A)$$

(HIGH ORDER DIFFERENCE)

5. $0 \rightarrow B01$

3609

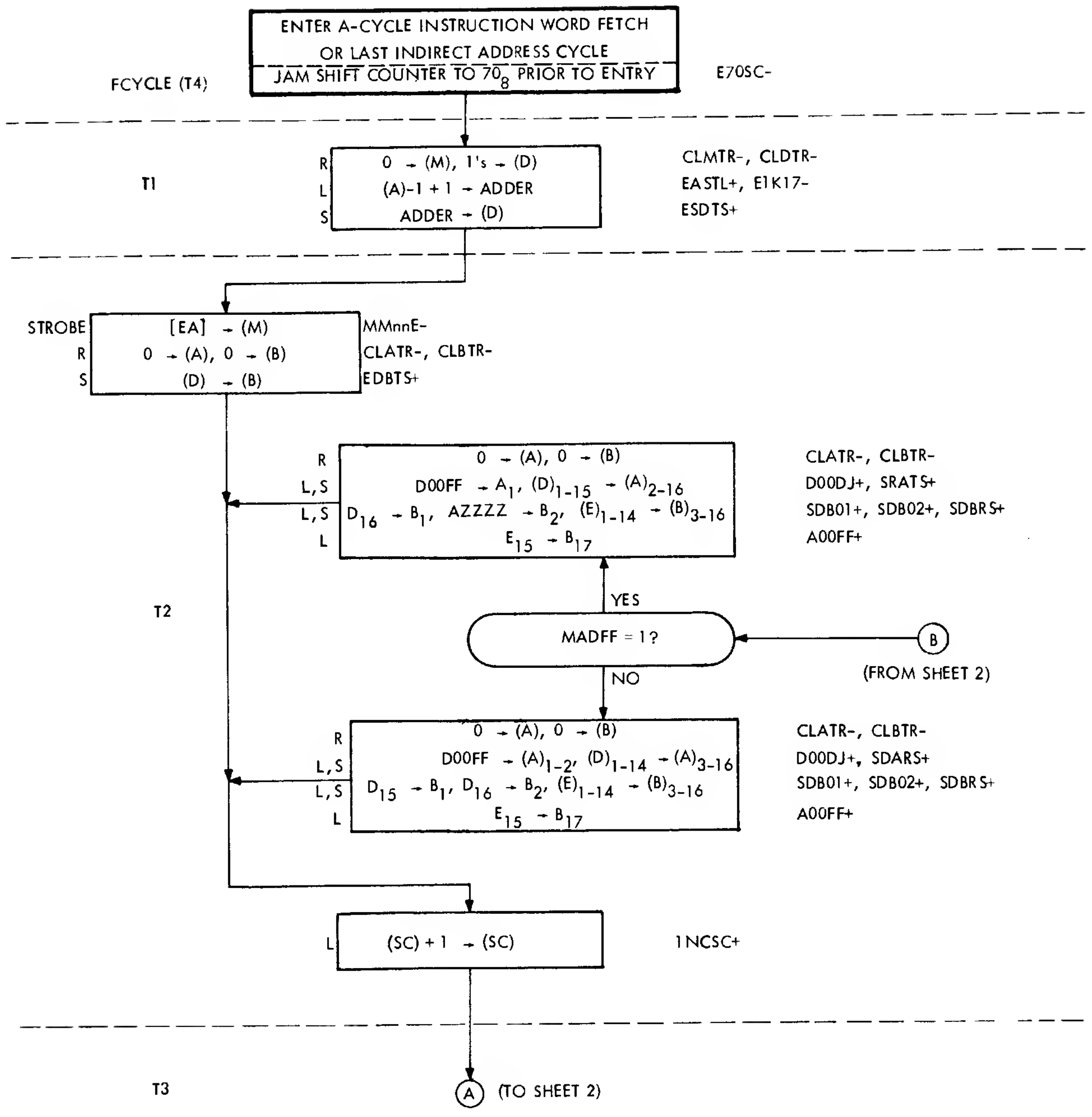
Figure 8. Double Subtract Simplified Diagram

List of Parts for DDP-516-11 High Speed Arithmetic Option

Reference Designation	Description	3C Part No.	Qty Req
A1C48	μ-PAC DIGITAL MODULE -- NAND Type 1 power amplifier	Model CC-045	1
A1B53, A1D53, 55, A1F52, 54	μ-PAC DIGITAL MODULE -- parallel transfer gate	Model CM-022	5
A1A36, A1A42, A1B46	μ-PAC DIGITAL MODULE -- multi-input NAND gate	Model DC-335	3
A1B42, A1B65	μ-PAC DIGITAL MODULE -- NAND gate Type 1	Model DI-335	2
A1A34, 38, A1B43, A1B66, A1C44, A1C67	μ-PAC DIGITAL MODULE -- NAND gate Type 2	Model DL-335	6
A1B36, A1B47, A1B58	μ-PAC DIGITAL MODULE -- expandable NAND gate	Model DN-335	3
A1B38	μ-PAC DIGITAL MODULE -- transfer gate NOTE These modules are mounted in the main frame logic drawer (A1-Unit), therefore, no additional connector planes are required.	Model TG-335	1

APPENDIX A
FLOW CHARTS / INSTRUCTION ANALYSES

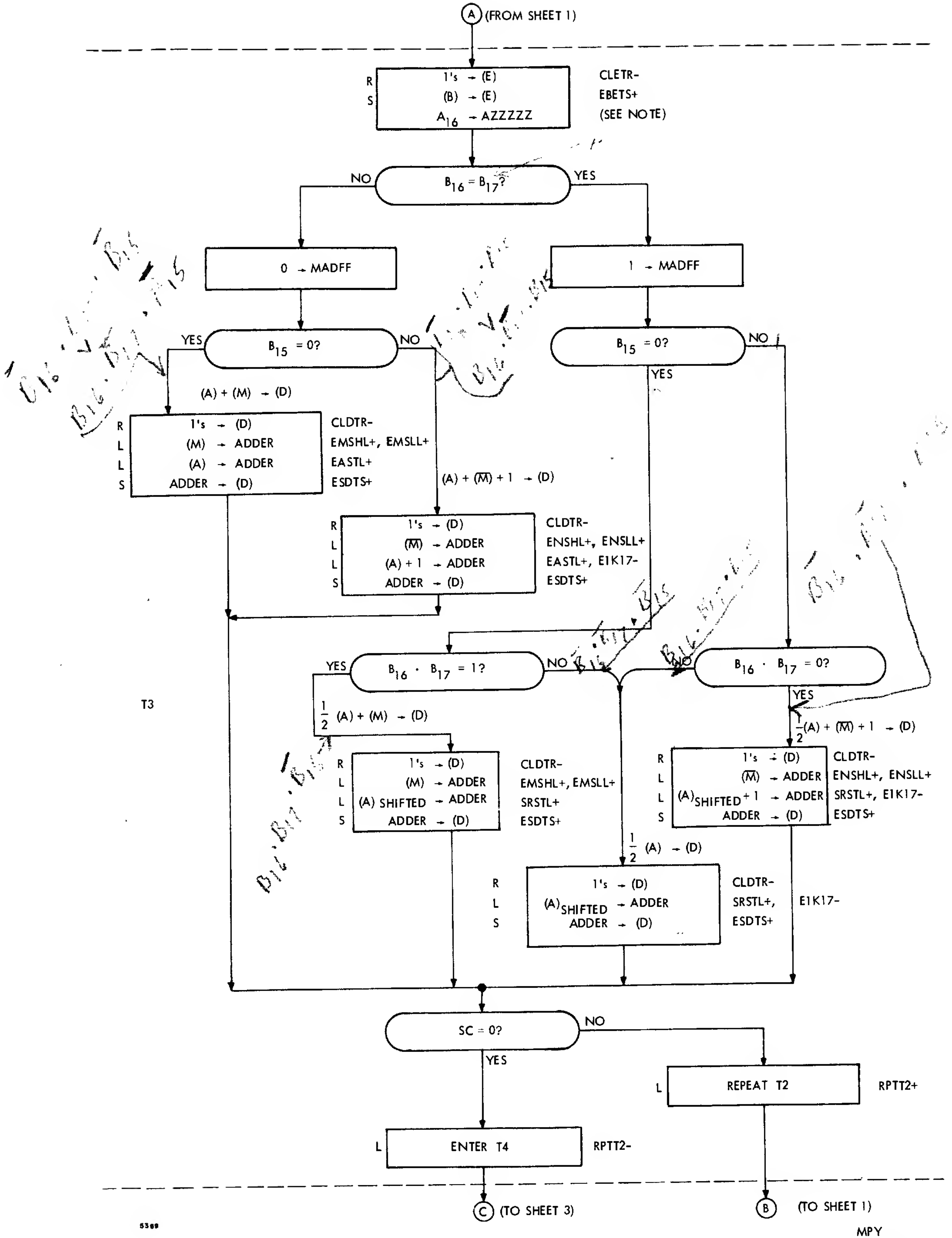
Title	Page
MPY Flow Chart	A-2
MPY Instruction	A-3
DIV Flow Chart	A-4
DIV Instruction	A-5
NRM Flow Chart	A-6
NRM Instruction	A-7
SCA Flow Chart	A-8
SCA Instruction	A-9
DBL Flow Chart	A-10
DBL Instruction	A-11
SGL Flow Chart	A-12
SGL Instruction	A-13
DLD Flow Chart	A-14
DLD Instruction	A-15
DST Flow Chart	A-16
DST Instruction	A-17
DAD Flow Chart	A-18
DAD Instruction	A-19
DSB Flow Chart	A-20
DSB Instruction	A-21



5366

MPY
5.5 CYCLES
OP CODE 16

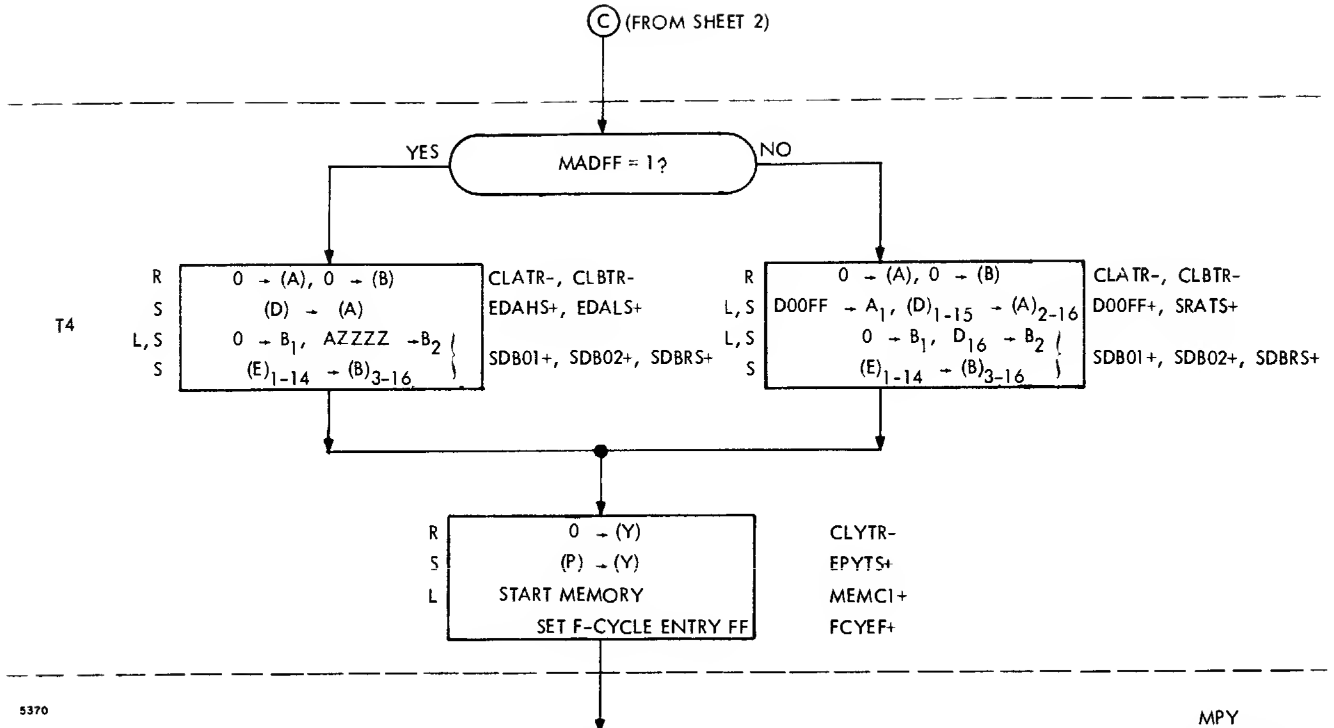
(SHEET 1 OF 3)



5369

MPY
5.5 CYCLES
OP CODE 16

(SHEET 2 OF 3)



5370

NOTE: MISSING SIGNALS CAN BE FOUND IN MPY ANALYSIS

MPY
5.5 CYCLES
OP CODE 16
(SHEET 3 OF 3)

Instruction: Multiply (MPY)

OP Code: 16   Type: MR, 5.5 Cycles

Description: (A) X [EA] → (A, B)

F	T	1	1	1	0	S	A	A	A	A	A	A	A	A	A
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Execution Time (μsec): 5.28

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
E70SC-	124-F3	F	TL4	L	(TL4FF+)(F01CY+)(MPY0P+)	124-F3	121-A2/J7	Set shift counter to 70_8
ACYEF+	119-F4	F	TL4	L	(M01FF-)(TL4FF+)(E01NS-)(F01CY+)	119-F4	119-H3	Set A-cycle
EASTL+	127-L1	A	TLATE-	L	(ACYEF+)(TLATE-)(TL1FF+)	127-J1	101--116-A5	Enable A-register to adder
CLMTR-	128-K8	A	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Reset M-register
CLDTR-	125-H5	A	TL1	R	(ACYEF+)(TL1FF+)(JST0P-)(1RS0P-)(1MA0P-)(MCRST+)	125-D4	101--116-E7	Clear D-register to ONEs
ESDTS+	125-L4	A	TL1	S	(ACYEF+)(TL1FF+)(JST0P-)(1RS0P-)(1MA0P-)(MCRST+)	125-D4	101--116-D4-D8	Enable adder sum to D-register
MMnnE-	153/160				(SWnnA+)(STRB1+)	153/160	101--116-H8	Memory data set into M-register
INCSC+	126-L3	A	TL2	L	(ACYEF+)(TL2FF+)(0PGMD+)	126-H5	121-A5	Enable increment shift counter
MDA2A-	123-E11	A	TL2	L	(ACYEF+)(TL1FF+)(MPY0P+)(SC14F-)(SC15F-)(SC16F-)	123-C2	122/123	Implement CLATR-, CLBTR-, EDBTS+
CLATR-	122-H7	A	TL2	R	(MDA2A+)(MCRST+)	122-F8	101--116-H5 124-K3	Clear A-register Reset A00FF
CLBTR-	123-J6	A	TL2	R	(MDA2A+)(MCRST+)	123-G5	101--116-H2	Clear B-register
EDBTS+	123-L1	A	TL2	S	(MDA2A+)(MCSET+)	123-J1	101--116-G3	Enable D-register to B-register
SRSTL-	128-H1				(MACYL+)∧[(B16FF+)(A00FF+)∨(B16FF-)(A00FF-)]	128-F1	124-H6 101--116-A3	B16 = B17
E1K17-	127-L4	A	TLATE	L	(MEMAC-)(SKGRP-)(TLATE+)(SUB0P-)(EYSLL-)(1RS0P-)(D1V0P-)	127-J6	116-D7-D9 117-B1	Force carry to adder
EASTL+	127-L1	A	TL3	L	(MACYL+)∧[(B16FF+)(A00FF-)∨(B16FF-)(A00FF+)]	127-E1	101--116-A5	Enable A-register to adder

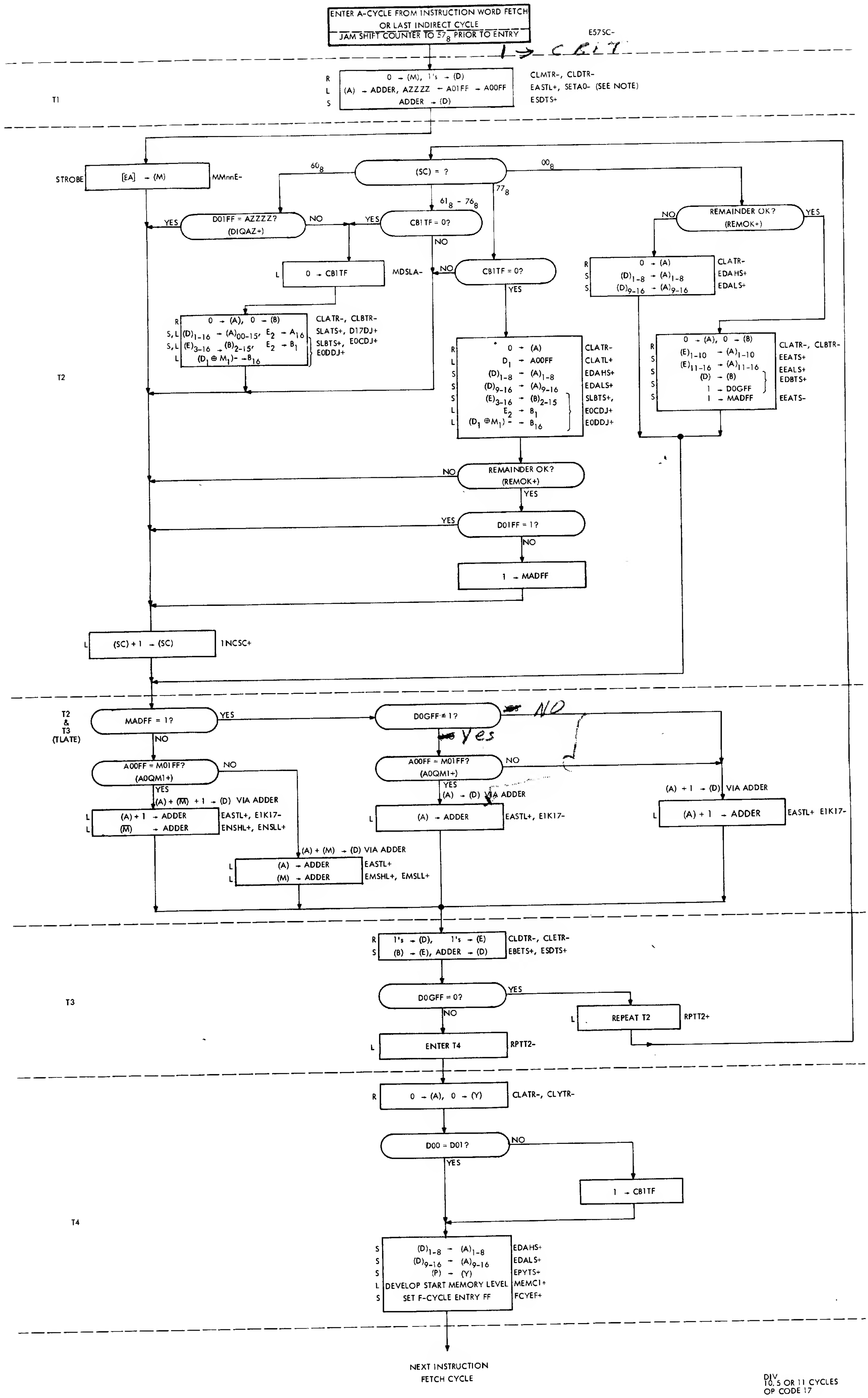
Instruction: (MPY)

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
EMSHL+	127-L8	A	TL3	L	(MACYL+)(B15FF-)∧[(B16FF+)∨(A00FF+)]	127-A10	101--107-A8	Enable M(1-7) to adder
EMSLL+	127-L10	A	TL3	L	(MACYL+)(B15FF-)∧[(B16FF+)∨(A00FF+)]	127-A10	108--116-A9	Enable M(8-16) to adder
ENSHL+	127-L7	A	TL3	L	[(MACYL+)(B15FF+)]∧[(B16FF-)∨(A00FF-)	127-C8	101--107-A9	Enable M-(1-7) to adder
ENSLL+	127-L5	A	TL3	L	(MACYL+)(B15FF+)∧[(B16FF-)∨(A00FF-)]	127-C8	108--116-A9	Enable M-(8-16) to adder
SETAZ+	125-J9	A	TL3	L	(ACYLF+)(TL3FF+)(MPY0P+)(A16FF+)	125-A9	125-L9 130-B7	Set AZZZZ FF
AZZZZ	125-L10	A	TL3	L	(A16FF-)(TL3FF+)(MPY0P+)	125-D2	125-L10	Reset AZZZZ FF
CLDTR-	125-J6	A	TL3	R	(ANA0P-)(TL3FF+)(MCRST+)	125-B6	101--116-E7	Clear D-register to ONEs
CLETR-	125-H2	A	TL3	R	(0PGMD+)(ACYLF+)(TL3FF+)(MCRST+)	123-B4	101--116-K3	Clear E-register to ONEs
MADFF-	124-L5	A	TL3	R	(MCRST+)(ACYLF+)(TL3FF+)(MPY0P+)	124-H5	See Wire List	MADFF reset
MADFF+	124-L6	A	TL3	S	(MCSET+)(TL3FF+)(SRSTL+)	124-H6	See Wire List	MADFF set
EBETS+	125-L1	A	TL3	S	(0PGMD+)(ACYLF+)(TL3FF+)(MCSET+)	123-B4	101--116-J2	Enable B-register to E-register
ESDTS+	125-L1	A	TL3	S	(TL3FF+)(10GRP-)(MCSET+)	125-D6	101--116-D6	Enable adder sum to D-register
D00DJ+	130-D1				(0PGMD+)(D00FF+)	130-B4	101-G6	D00FF into A_1
SDB01+	130-B5				(MADFF-)(D15FF-) or (MADFF+)(D16FF-)	130-B5 130-B6	101-D1	D15FF into B_1 D16FF into B_1
SDB02+	130-B7				(MADFF-)(D16FF-) or (MADFF+)(AZZZZ-)	130-B7 130-B8	102-D1	D16FF into B_2 Set B-register bit 2
CLATR-	122-H7	A	TL2	R	(MADFF+)(TL2FF+)(MPY0P+)(MCRST+) or (MADFF-)(TL2FF+)(MPY0P+)(SCQ70-)(ACYEF+)(MCRST+)	123-E10/122-F8 123-E11/122-F8	101--116-H5	Clear A-register
CLBTR-	123-J6	A	TL2	R	(MADFF+)(TL2FF+)(MPY0P+)(MCRST+) or (MADFF-)(TL2FF+)(MPY0P+)(SCQ70-)(ACYEF+)(MCRST+)	123-E10 123-E11	101--116-H2	Clear B-register
SRATS+	122-L11	A	TL2	S	(MDSRA+)(MCSET+) *	122-H11	101--116-G6	Shift right A-register
SDBRS+	123-L10	A	TL2	S	(MDSRA+)(MCSET+)	123-G10	101--116-D1 123-F4	Double shift right B-register Enable set A00FF
SDARS+	123-L11	A	TL2	S	(MDA2C+)(MCSET+)**	123-J11	101--116-D2	Double shift right A-register
CLATR-	122-H7	A	TL4	R	(MDG4D-)(MCRST+) or (MDSRA+)(MCRST+)***	122-F7 122-F7	101--116-H5	Clear A-register
CLBTR-	123-J6	A	TL4	R	(MDG4D+)(MCRST+) or (MDSRA+)(MCRST+)	123-G8	101--116-H2	Clear B-register
MEMC1+	126-J11	A	TL4	L	(TL4FF+)(SPM0D-)(TLAFF-)	126-F11/H11	150-C1	Enable set RCYF1+
CLYTR-	129-J3	A	TL4	R	(ACYNX-)(TL4FF+)(MCRST+)	129-D3	101--116-L11	Clear Y-register
SDBRS+	123-L10	A	TL4	S	(MDG4D+)(MCSET+) or (MDSRA+)(MCSET+)	123-G10	101--116-D1	Double shift right B-register
EDAHS+	122-L1	A	TL4	S	(MDG4D+)(MCSET+)	122-H1	101--108-G7	Enable D(1-8) into A(1-8)
EDALS+	122-L2	A	TL4	S	(MDG4D+)(MCSET+)	122-H1	109--116-G7	Enable D(8-16) into A(8-16)
SRATS+	122-L11	A	TL4	S	(MDSRA+)(MCSET+)	122-H11	101--116-G6	Shift right A-register
EPYTS+	129-L4	A	TL4	S	(P1SEX-)(E01NS+)(0PGJS-)(MCSET+)	129-D3	101--116-K11	Enable P-register to Y register
RCYF1+	150-D1	A	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle

*See 123-E10 for MDSRA-

**See 123-E11 for MDA2C-

***See 123-E7 for MDG4D- and 123-E9 for MDSRA-



DIV
10, 5 OR 11 CYCLES
OP CODE 17

NOTE: MISSING SIGNALS CAN BE FOUND IN DIV ANALYSIS

<u>Instruction</u>: Divide (DIV)

<u>OP Code</u>: 17 <u>Type</u>: MR, 10.5 or 11 Cycles

<u>Description</u>: (A, B) ÷ [EA] → (A, B)
OVF → (C)

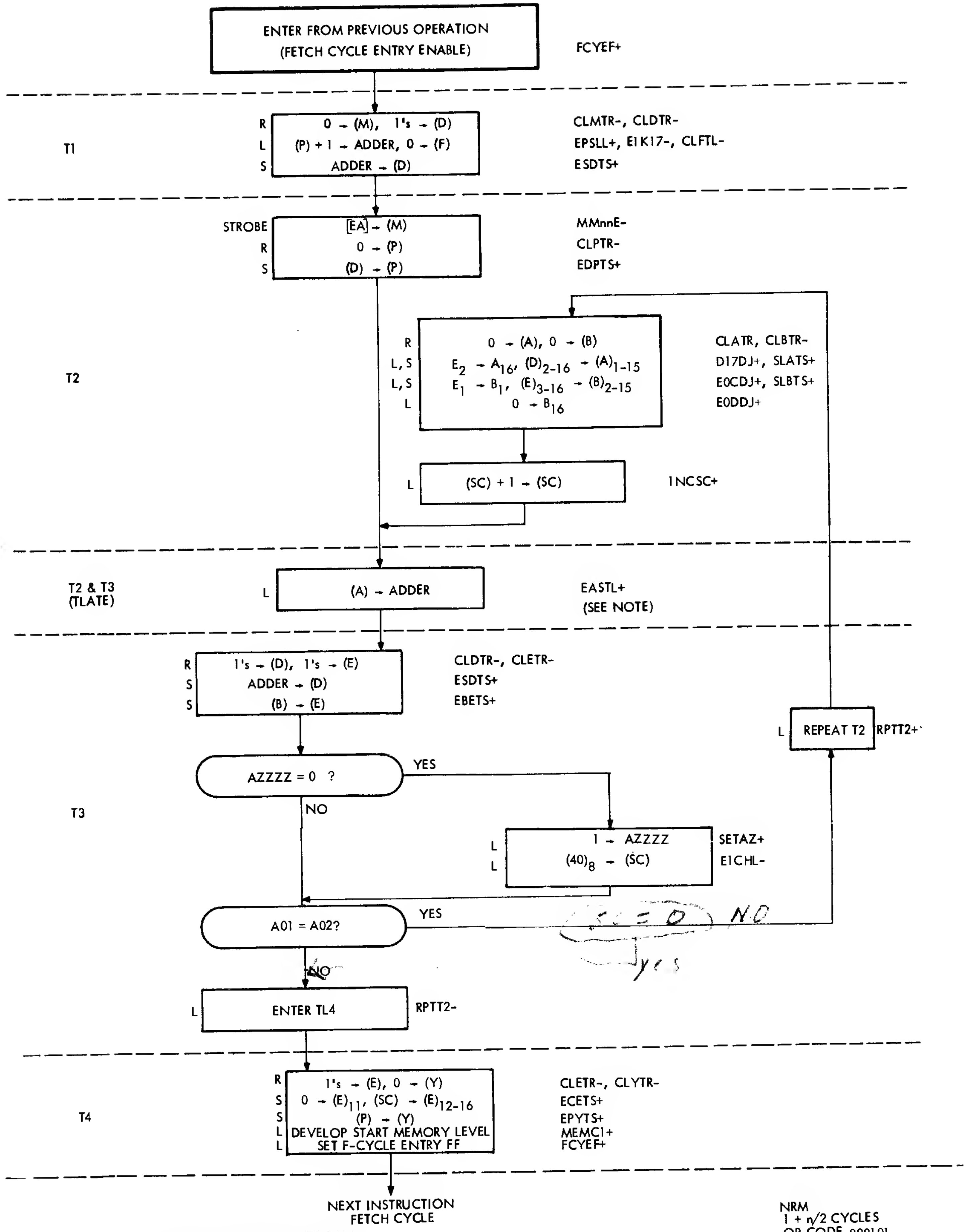
F	T	1	1	1	1	S	A	A	A	A	A	A	A	A	A
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Execution Time (μsec): 10.08 or 10.56

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
E57SC-	124-J3	F	TL4	L	(TL4FF+)(F01CY+)(D1V0P+)	124-J3	124-L3 124-F2 121-B7/E7/D7	Reset A00FF Set CB1TF Set shift counter to 57_8
ACYEF+	119-F4	F	TL4	L	(M01FF+)(TL4FF+)(E01NS-) (F01CY+)	119-F4	119-H3	Set A-cycle at next TL1
CLMTR-	128-K8	A	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Reset M-register
CLDTR-	125-H5	A	TL1	R	(ACYEF+)(TL1FF+)(JST0P-) (1RS0P-)(1MA0P-)(MCRST+)	125-D4	101--116-E7	Clear D-register to ONEs
EASTL+	127-L1	A	TL1	L	(ACYEF+)(TLATE-)(CAS0P-) (LSX0P-)(10GRP-)	127-J1	101--116-A5	Enable A-register to adder (Don't Care operation)
SETA0-	125-J9	A	TL1	L	(ACYEF+)(TL1FF+)(D1V0P+) (A01FF+)	125-D9	125-H9 124-K4	A01FF into AZZZZ FF A01FF into A00FF
ESDTS+	125-L4	A	TL1	S	(ACYEF+)(TL1FF+)(JST0P-) (1RS0P-)(1MA0P-)(MCSET+)	125-D4	101--116-D4-D8	Enable adder sum to D-register (Don't Care operation)
MMnnE-	153/160				(SWnnA+)(STRB1+)	153/160	101--116-H8	Memory data set into M-register
A0QM1+	124-H7	A	TL2	L	(A00FF+)(M01FF+)V (A00FF-)(M01FF-)	124-F7	127-A8 127-C7	A00FF equals M01FF
INCSC+	126-H5	A	TL2	L	(ACYEF+)(TL2FF+)(0PGMD+)	126-H5	121-A5	Increment shift counter
EMSHL+	127-L8	A	TLATE	L	(D1V0P+)(ACYLF+)(TLATE+) (MADFF-)(A0QM1-)	127-A8	101--107-A8	Enable M(1-7) to adder
EMSLL+	127-L10	A	TLATE	L	(D1V0P+)(ACYLF+)(TLATE+) (MADFF-)(A0QM1-)	127-A8	108--116-A8	Enable M(8-16) to adder
EASTL+	127-L1	A	TLATE	L	(D1V0P-)(TLATE+)(ACYLF+)	127-C1	101--116-A5	Enable A-register to adder
ENSHL+	127-L7	A	TLATE	L	(ACYLF+)(TLATE+)(D1V0P+) (MADFF-)(A0QM1+)	127-C7	101--107-A9	Enable M-(1-7) to adder
ENSLL+	127-L5	A	TLATE	L	(ACYLF+)(TLATE+)(D1V0P+) (MADFF-)(A0QM1+)	127-C7	108--116-A9	Enable M-(8-16) to adder
E1K17-	127-J6	A	TLATE	L	(A0QM1+)(ACYEF+)(0PGMD+)	127-C4	116-D7/D9 117-B1	Force carry to adder
CLETR-	125-J2	A	TL3	R	(0PGMD+)(ACYLF+)(TL3FF+) (MCRST+)	125-A4	101--116-L3	Clear E-register to ONEs
CLDTR-	125-H5	A	TL3	R	(ANA0P-)(TL3FF+)(MCRST+)	125-H5	101--116-E7	Clear D-register to ONEs
EBETS+	125-L1	A	TL3	S	(0PGMD+)(ACYLF+)(TL3FF+) (MCSET+)	125-A4	101--116-J2	Enable B-register to E-register
ESDTS+	125-L4	A	TL3	S	(ANA0P-)(TL3FF+)(MCSET+)	125-H4	101--116-D4-D8	Enable adder sum to D-register
RPTT2+	126-F5	A	TL3	L	(ACYLF+)(D1V0P+)(D0GFF-)	126-D5	118-A3	Repeat TL2 timing level
D1QAZ-	123-B4	A	TL2	L	(D01FF+)(AZZZZ+)V(D01FF-) (AZZZZ-)	123-A3	123-A1 122-A9	D01FF equals AZZZZFF
MDSLA-	122-B8	A	TL2	L	(D1V0P+)(TL2FF+)(D1QAZ-) (SC16FF-)(SC15FF-) (SC14FF-)(SC13FF-) (SC12FF+)	122-B9	122-F9 123-G5 124-F1	Enable shift left A- and B-register Reset CB1TF on reset clock
CLATR-	122-H7	A	TL2	R	(MDSLA+)(MCRST+)	122-F9	101--116-H5 124-K3	Clear A-register Clear A00FF
CLBTR-	123-J6	A	TL2	R	(MDSLA+)(MCRST+)	123-G5	101--116-H2	Clear B-register
SLATS+	122-L10	A	TL2	S	(MDSLA+)(MCSET+)	122-L10	101--116-G5	Shift left A-register
D17DJ+	130-F3	A	TL2	L	(0PGMD+)(E02FF+)	130-D5	116-G5	Enter E02FF into A16FF
SLBTS+	123-L4	A	TL2	S	(MDSLA+)(MCSET+)	123-J4	101--116	Shift left B-register
E0CDJ+	130-D8	A	TL2	L	(0PGMD+)(E02FF-)	130-D6	101-E1	Enter E02FF into B01FF
E0DDJ+	130-D11	A	TL2	L	(D1V0P+)Λ(M01FF+) (D01FF+)V(M01FF-) (D01FF-)	130-B10	116-G1	Set B16FF if D01FF = M01FF
MDSLA-	122-B8	A	TL2	L	(D1V0P+)(TL2FF+)(SCZR0-) (B1TFF-)(SCQ77-)	122-D8	122-F9 123-G5 124-F1	Enable shift left A- and B-register Reset CB1TF on reset clock

<u>Instruction</u>: (DIV)

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
CLATL+	122-F7	A	TL2	L	(D1V0P+)(TL2FF+)(SCZR1+) (MDG2E-)(ACYLF+)	122-A6	122-H7 124-H4	Enable clear A-register Enable D01FF into A00FF
CLATR-	122-H7	A	TL2	R	(CLATL+)(MCRST+)	122-H7	101--116-H5	Clear A-register
EDAHS+	122-L1	A	TL2	S	(CLATL+)(MCSET+)	122-H1	101--108-G7	Enable D(1-8) into A(1-8)
EDALS+	122-L2	A	TL2	S	(CLATL+)(MCSET+)	122-H1	109--116-G7	Enable D(9-16) into A(9-16)
REM0K+	123-B1	A	TL2	L	(DG0NE-) V (D1QAZ-)	123-A1	123-C1 124-F6	Remainder OK
MADFF	124-L6	A	TL2	S	(MCSET+)(TL2FF+)(ACYEF+) (D1V0P+)(REM0K+)(SCZR1+) (D01FF-)	124-F5	See Wire List	MADFF set
MDG2E-	123-D1	A	TL2	L	(REM0K+)(D1V0P+)(TL2FF+) (E01NS+)	123-C1	122-A6/F3 123-G1 124-L8	Initiate terminate divide
D0GFF+	124-L8	A	TL2	L	(MDG2E-)	124-L8	See Wire List	D0GFF set
CLATL+	122-F7	A	TL2	L	(MDG2E-)	122-F9	122-H7 124-H4	Enable clear A-register Enable D01 into A00FF
CLATR-	122-H7	A	TL2	R	(CLATL+)(MCRST+)	122-H7	101--116-H5	Clear A-register
CLBTR-	123-J7	A	TL2	R	(MDG2E+)(MCRST+)	123-G5	101--116-H2	Clear B-register
EEATS+	122-L4	A	TL2	S	(MDG2E+)(MCSET+)	122-H3	101--110-G4	Enable E(1-10) into A(1-10)
EEALS+	122-K4	A	TL2	S	(EEATS-)	122-K4	111--116-G4	Enable E(11-16) into A(11-16)
EDBTS+	123-L1	A	TL2	S	(MDG2E+)(MCSET+)	123-J1	101--116-G3	Enable D-register into B-register
EMSHL+	127-L8	A	TLATE	L	(A0QM1-)(ACYEF+)(D1V0F+) (D0GFF+)	127-C5	101--107-A8	Enable M(1-7)to adder
ENSHL+	127-L7	A	TLATE	L	(A0QM1-)(ACYEF+)(D1V0P+) (D0GFF+)	127-C5	101--107-A9	Enable M-(1-7)to adder
EMSLL+	127-L10	A	TLATE	L	(A0QM1-)(ACYEF+)(D1V0P+) (D0GFF+)	127-C5	108--116-A8	Enable M(8-16)to adder
ENSLL+	127-L5	A	TLATE	L	(A0QM1-)(ACYEF+)(D1V0P+) (D0GFF+)	127-C5	108--116-A9	Enable M-(8-16)to adder
E1K17+	127-L4	A	TLATE	L	(A0QM1-)(ACYEF+)(D1V0P+) (D0GFF+)	127-C5	116-D7	Force carry to adder
CLATR-	122-H7	A	TL4	R	(D1V0P+)(TL4FF+)(ACYLF+) (MCRST+)	122-A7	101--116-H5	Clear A-register
EDAHS+	122-L1	A	TL4	S	(D1V0P+)(TL4FF+)(ACYLF+) (MCSET+)	122-H1	101--108-G7	Enable D(1-8) to A(1-8)
EDALS+	122-L2	A	TL4	S	(D1V0P+)(TL4FF+)(ACYLF+) (MCSET+)	122-H1	109--116-G7	Enable D(9-16)to A(9-16)
CB1TF+	124-L2	A	TL4	S	(D1V0P+)(D00 ≠ D01) (TL4FF+)(MCSET+)	124-D2	124-L2	CB1TF set
CLYTR-	128-J3	A	TL4	R	(SCZR0-)	129-A1	101--116-G7	Clear Y-register
EPYTS+	129-L5	A	TL4	S	(P1SEX-)(E01NS+)(0PGJS-)	129-D4/H4	101--116-K11	Enable P-register to Y-register
MEMC1+	126-J11	A	TL4	L	(TL4FF+)(SPM0D-)(TLAFF-)	126-F11 H11	150-C1	Enable set RCYF1+
RCYF1+	150-D1	A	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle



NRM
1 + n/2 CYCLES
OP CODE 000101

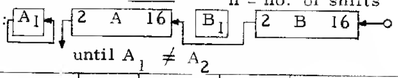
3612

NOTE: MISSING ADDER INPUTS CAN BE FOUND IN NRM ANALYSIS

Instruction: Normalize (NRM)

OP Code: 000101 Type: 1 + n/2 cycles n = no. of shifts

Description:

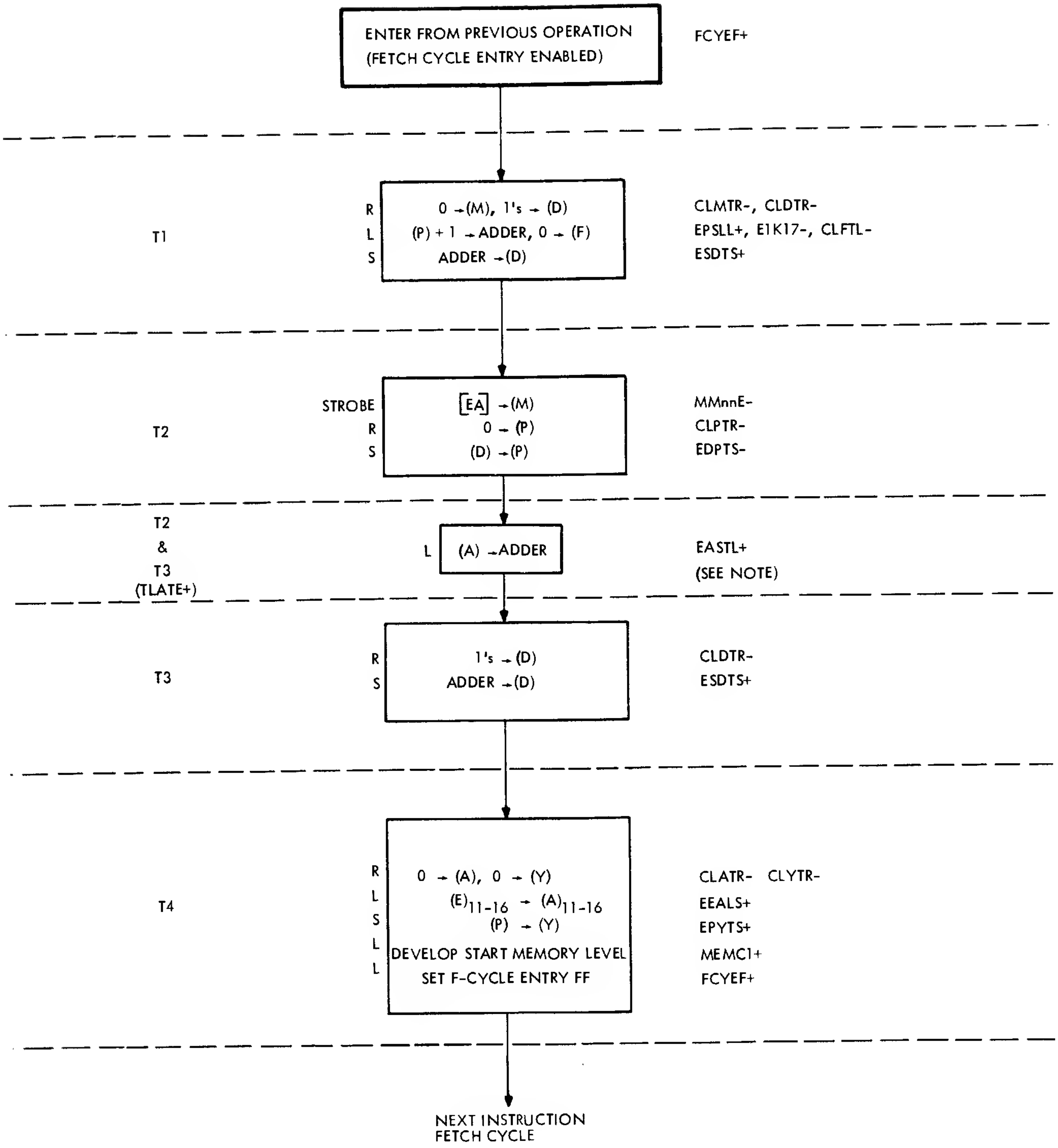


0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Execution Time (μsec): 0.96 + 0.48n

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
CLMTR-	128-K8	F	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Clear M-register
CLDTR-	125-J5	F	TL1	R	(1CYEF-)(ACYEF-)(TL1FF+)(MCRST+)	125-A5	101--116-E7 130-H4	Clear D-register to ONEs Set D00FF
CLFTL-	125-J7	F	TL1	L	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	120-B1 121-A5 125-L10	Clear F-register Clear shift counter Reset AZZZZ FF
EPSLL+	128-H4	F	TL1	L	(FCYEF+)(TLATE-)	128-F3	101--116-A10	Enable P-register to adder
E1K17-	127-L4	F	TL1	L	(JAMKN-)	127-J5	116-D7-D9	Force carry to adder
ESDTS+	125-L4	F	TL1	S	(1CYEF-)(ACYEF-)(TL1FF+)(MCSET+)	125-A5	101--116-D4-D8	Enable adder sum to D-register
MMnnE-	153/160				(SWnnA+)(STRB1+)	153/160	101--116-H8	Memory data set into M-register
CLPTR-	129-J10	F	TL2	R	(EDPTL+)(MCRST+)*	129-H9	101--116-H10	Clear P-register
EDPTS+	129-L9	F	TL2	S	(EDPTL+)(MCSET+)	129-H9	101--116-C11	Enable D-register into P-register
EASTL+	127-L1	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G11	101--116-A5	Enable A-register to adder
EMSHL+	127-L8	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G11	101--107-A8	Enable M(1-7) to adder
ENSHL+	127-L7	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G11	101--107-A9	Enable M-(1-7) to adder
EMSLL+	127-L10	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G11	108--116-A8	Enable M(8-16) to adder
ENSLL+	127-L5	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G11	108--116-A9	Enable M-(8-16) to adder
CLDTR-	125-J5	F	TL3	R	(TL3FF+)(ACYLF-)(MCRST+)	125-D6	101--116-E7	Clear D-register to ONEs
CLFTR-	127-J2	F	TL3	R	(M01FF+)(TL3FF+)(GEN0P+)(M01FF-)(MCRST+)	125-D1	101--116-L3	Clear E-register to ONEs
ESDTS+	125-L4	F	TL3	S	(TL3FF-)(10GRP-)(MCSET+)	125-D6	101--116-D4-D8	Enable adder sum to D-register
EBETS+	125-L1	F	TL3	S	(M01FF+)(TL3FF+)(GEN0P+)(M01FF-)(MCSET+)	125-D1	101--116-J2	Enable B-register to E-register
SETAZ+	125-J9	F	TL3	L	(E1CHL-)	125-F10	125-L10	Set AZZZZ FF
E1CHL-	125-F10	F	TL3	L	(A1QAZ+)(NRM0P+)(TL3FF+)(AZZZZ-)	125-F10	125-J9 121-A2 126-F5	SETAZ+ Set shift counter to (40_8) RPTT2+
RPTT2+	126-F4	F	TL3	L	(E1CHL-)V(FCYLF+)(A1QAZ+)(SCZR0-)	126-B3/D3/F4	118-A3	Repeat TL2
CLATR-	122-H7	F	TL2	R	(NRM0P+)(TL2FF+)(AZZZZ+)(MCRST+)	122-D9	101--116-H5	Clear A-register and generate MDSLA-
CLBTR-	123-J7	F	TL2	R	(MDSLA-)(MCRST+)	123-G5	101--116-H2	Clear B-register
D17DJ+	130-F3	F	TL2	L	(M08FF-)(M10FF+)(E02FF+)	130-D4	116-G5	Left shift end effect
SLATS+	122-L10	F	TL2	S	(NRM0P+)(TL2FF+)(AZZZZ+)(MCSET+)	122-D9/H10	101--116-G5	Shift left A-register
E0CDJ+	130-D8	F	TL2	L	(ACYLF-)(M10FF+)(E01FF-)	130-D7	101-F1	Set E_1 into B_1
SLBTS+	123-L4	F	TL2	S	(MDSLA-)(MCSET+)	123-J6	101--116-F1	Shift left B-register
E0DDJ-	130-D11	F	TL2	L	(M09FF+)-	130-B11	116-G1	Clear B_{16}
1NCSC+	126-L3	F	TL2	L	(FCYEF+)(TL2FF+)	126-H3	121-A5	Increment shift counter
A1QA2+	126-B3	F	TL3	L	(NRM0P+)Λ(A01FF+)(A02FF-)V(A01FF-)(A02FF+)	126-B4	126-D4	A-register bit 1 equals bit 2
CLETR-	125-H2	F	TL4	R	(NRM0P+)(TL4FF+)(MCRST+)	125-D3	101--116-L3	Clear E-register to ONEs
ECETS+	125-	F	TL4	S	(NRM0P+)(TL4FF+)(MCSET+)	125-D3	111-116-L5	Shift counter 11-16 into E-register 11-16
CLYTR-	129-J3	F	TL4	R	(ACYNX-)(TL4FF+)(MCRST+)	129-D3	101--116-L11	Clear Y-register
EPYTS+	129-L4	F	TL4	S	(P1SEX-)(E01NS+)(0PGJS-)(TL4FF+)(MCSET+)	129-D4/H4	101--116-K11	Enable P-register to Y-register
MEMC1+	126-J11	F	TL4	L	(TL4FF+)(SPM0D-)(TLAFF-)	126-F11/H11	150-C1	Enable set RCYF1+
RCYF1+	150-D1	F	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle

* See gate A1A44-F at 129-D8 for EDPTL+



3610

NOTE: MISSING SIGNALS CAN BE FOUND IN SCA ANALYSIS

SCA
1 CYCLE
OP CODE 000041

Instruction: Shift Count to A (SCA)

OP Code: 000041 Type: G, 1 cycle

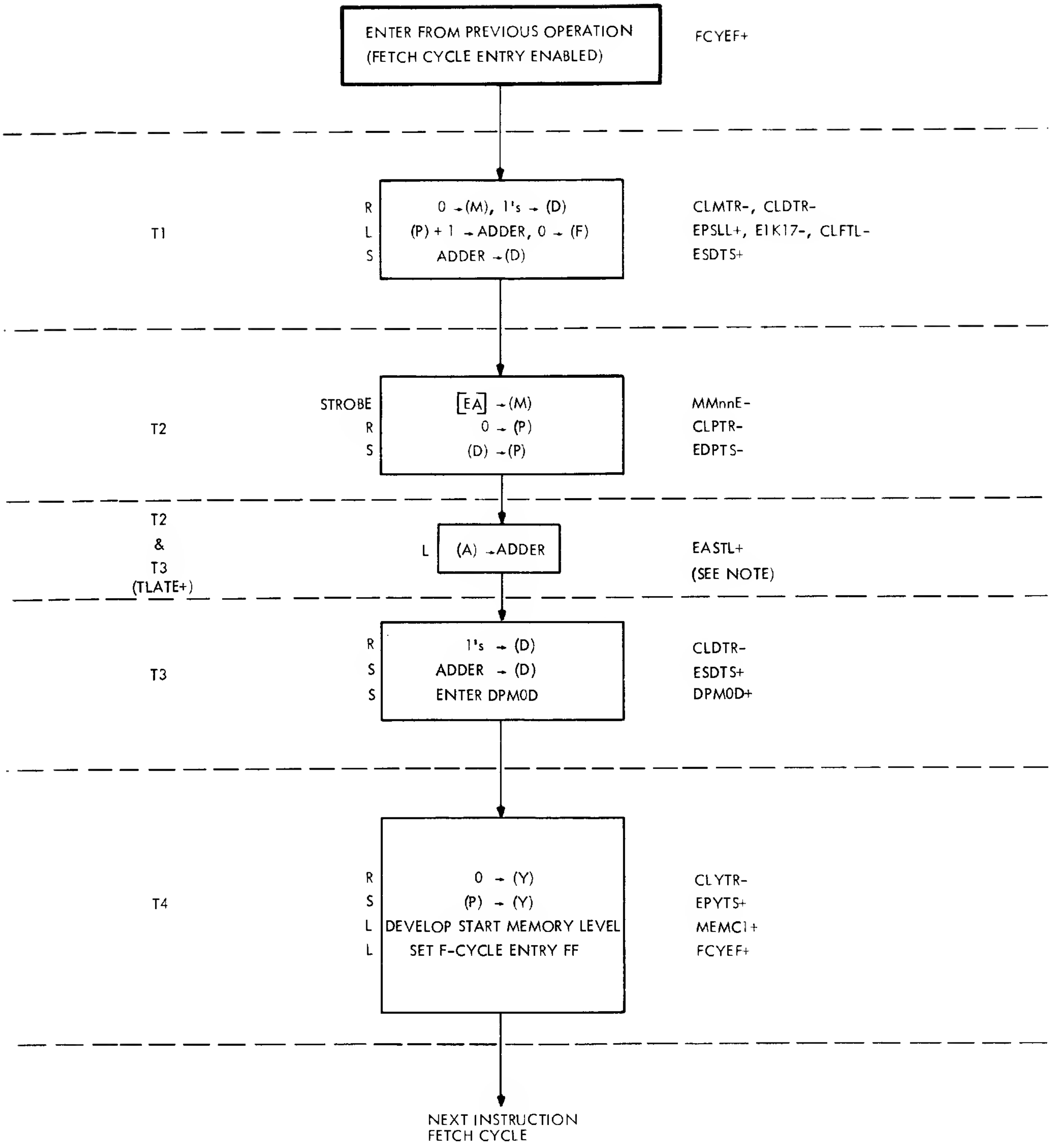
Description: $(SC)_{11-16} \rightarrow (A)_{11-16}$, $0 \rightarrow (A)_{1-10}$

0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Execution Time (μsec)· 0. 96

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
CLMTR-	128-K8	F	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Clear M-register
CLDTR-	125-J5	F	TL1	R	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	101--116-E7	Clear D-register to ONEs
EPSLL+	128-H4	F	TL1	L	(FCYEF+)(TLATE-)	128-F3	101--116-A10	Enable P-register to adder
E1K17-	127-L4	F	TL1	L	(JAMKN-)	127-J5	116-D7-D9	Force carry to adder
JAMKN-	127-J5	F	TL1	L	[(TLATE+)(ACYEF+)]	127-C3	127-L4	Implement E1K17-
CLFTL-	125-J7	F	TL1	L	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	120-B1 121-A5 125-L10	Clear F-register Clear shift counter Clear AZZZZ FF
ESDTS+	125-L4	F	TL1	S	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	101-116-D4-D8	Enable adder sum to D-register
MMnnE-	153/160				(SWnnA+)(STRB1+)	153/160	101--116-H8	Memory data set into M-register
CLPTR-	129-J10	F	TL2	R	(EDPTL+)(MCRST+)*	129-H10	101--116-H10	Clear P-register
EDPTS+	129-L9	F	TL2	S	(EDPTL+)(MCSET+)	129-H9	101--116-G11	Enable D-register into P-register
EASTL+	127-L1	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	129-G9	101--116-A5	Enable A-register to adder
EMSHL+	127-L8	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	101--108-A8	Enable M(1-7) to adder
ENSHL+	127-L7	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	101--107-A9	Enable M-(1-7) to adder
EMSLL+	127-L10	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	108--116-A8	Enable M(8-16) to adder
ENSLL+	127-L5	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	108--116-A9	Enable M-(8-16) to adder
(E1K17+)	127-J6	F	TLATE	L	See Gate A1B55-E	127-J6	116-D7-D9 117-B1	Jam carry network
CLDTR-	127-J5	F	TL3	R	(TL3FF+)(ACYLF-)	125-D6	101--116-E7	Clear D-register to ONEs
ESDTS+	125-L4	F	TL3	S	(TL3FF+)(10GRP-)	125-D6	101--116-D4-D8	Enable adder sum to D-register
CLATR-	122-H7	F	TL4	R	(GEN0B+)(M11FF+)(TL4FF+)(MCRST+)	122-D4	101--116-H5	Clear A-register
EEALS+	122-J4	F	TL4	L	(GEN0B+)(M11FF+)(TL4FF+)	122-D4	111-116-G4	Enable E(11-16) into A(11-16)
CLYTR-	129-J3	F	TL4	R	(ACYNX-)(TL4FF+)(MCRST+)	128-D3	101--116-L11	Clear Y-register
EPYTS+	129-L4	F	TL4	S	(P1SEX-)(E01NS+)(0PGJS-)	129-D4/H4	101--116-K11	Enable P-register to Y-register
MEMC1+	126-J11	F	TL4	L	(TL4FF+)(SPM0D-)(TLAFF-)	126-F11/H11	150-C1	Enable set RCYF1+
RCYF1+	150-D1	F	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle

*See gate A1A44-F at 129-D8 for EDPTL+



NOTE: MISSING SIGNALS CAN BE
FOUND IN DBL ANALYSIS

DBL
1 CYCLE
OP CODE 000007

3606

Instruction: Enter Double-Precision Mode (DBL)

OP Code: 000007 Type: G, 1 cycle

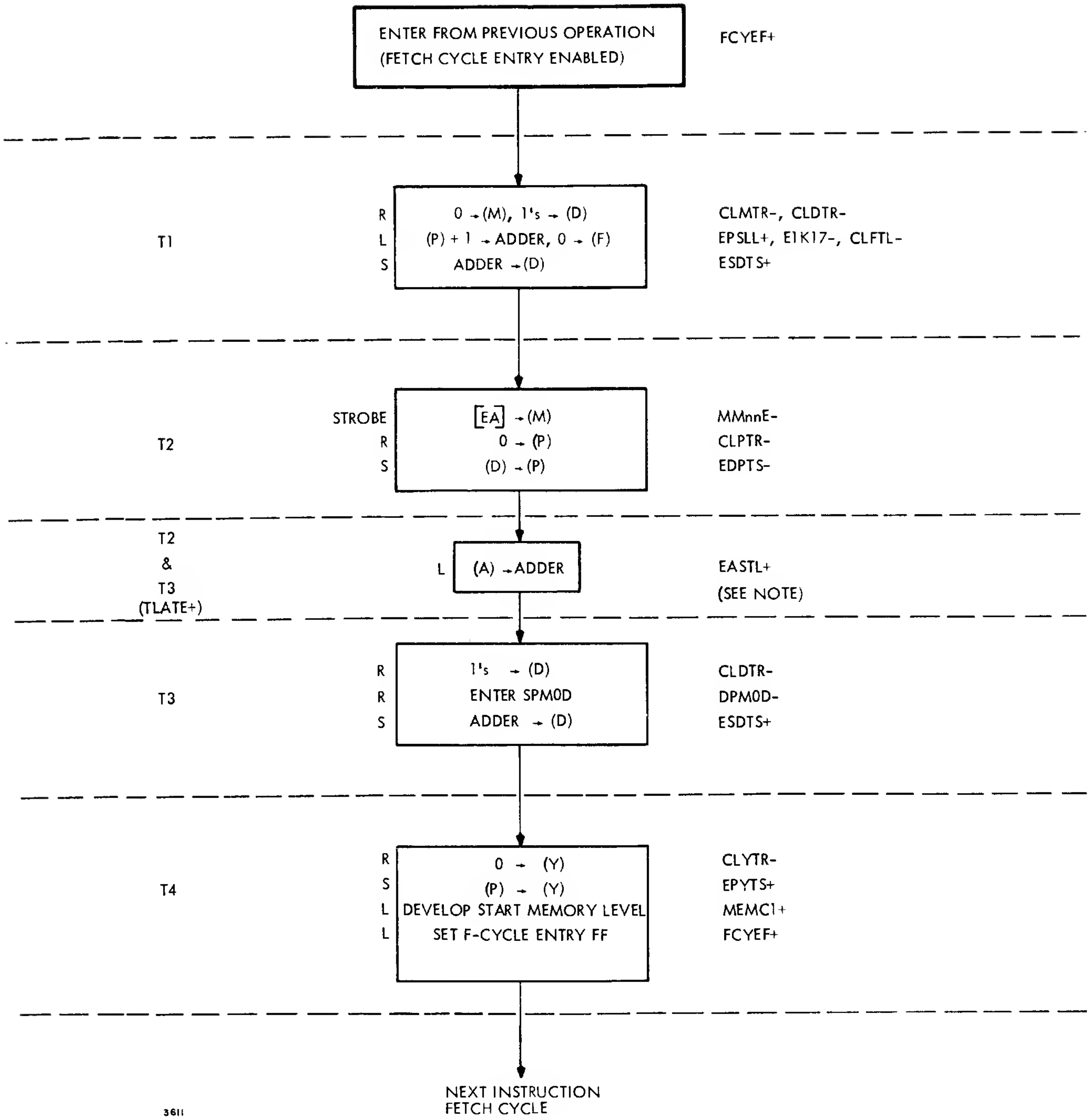
Description: Enter DBL for LDA, STA, ADD, and SUB

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Execution Time (μsec): 0. 96

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
EPSLL+	128-H4	F	$\overline{\text{TLATE}}$	L	(FCYEF+)(TLATE-)	128-F3	101--116-A10	Enable P-register to adder
E1K17-	127-L4	F	$\overline{\text{TLATE}}$	L	(TLATE-)	127-J5	116-D7-D9	Force carry to adder
CLMTR-	128-K8	F	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Clear M-register
CLDTR-	125-J5	F	TL1	R	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	101--116-E7	Clear D-register to ONEs
CLFTL-	125-J7	F	TL1	L	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	120-B1 121-A5 125-L10	Clear F-register Clear shift counter Clear AZZZZ FF
ESDTS+	125-L4	F	TL1	S	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	101--116-D4-D8	Enable adder sum to D-register
MMnnE-	153/160				(SWnnA+)(STRB1+)	153/160	101--116-H8	Memory data set into M-register
CLPTR-	129-J10	F	TL2	R	(EDPTL+)(MCRST+)*	129-H10	101--116-H10	Clear P-register
EDPTS+	129-L9	F	TL2	S	(EDPTL+)(MCSET+)	129-H9	101--116-G11	Enable D-register into P-register
EASTL+	127-L1	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	129-G9	101--116-A5	Enable A-register to adder
EMSHL+	127-L8	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	101--108-A8	Enable M(1-7) to adder
ENSHL+	127-L7	F	TLATE	L	(GEN0P+) (TLATE+)(M01FF-)	127-G9	101--107-A9	Enable M-(1-7) to adder
EMSLL+	127-L10	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	108--116-A8	Enable M(8-16) to adder
ENSLL+	127-L5	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	108--116-A9	Enable M-(8-16) to adder
(E1K17+)	127-J6	F	TLATE	L	See gate A1B55-E	127-J6	116-D7-D9 117-B1	Jam carry network
CLDTR-	127-J5	F	TL3	R	(TL3FF+)(ACYLF-)	125-D6	101--116-E7	Clear D-register to ONEs
ESDTS+	125-L4	F	TL3	S	(TL3FF+)(10GRP-)	125-D6	101--116-D4-D8	Enable adder sum to D-register
DPM0D+	124-B10	F	TL3	S	(GEN0B+)(TL3FF+)(M15FF+)(M14FF+)(MCSET+)	124-B8	102-H6 123-E3 125-B2 127-A4/A6 132-F8	Enable double precision operations
CLYTR-	129-J3	F	TL4	R	(ACYNX-)(TL4FF+)(MCRST+)	129-D3	101--116-L11	Clear Y-register
EPYTS+	129-L4	F	TL4	S	(P1SEX-)(E01NS+)(0PGJS-)	129-D4/H4	101--116-K11	Enable P-register to Y-register
MEMC1+	126-J11	F	TL4	L	(TL4FF+)(SPM0D-)(TLAFF-)	126-F11/H11	150-C1	Enable set RCYF1+
RCYF1+	150-D1	F	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle

*See gate A1A44-F at 129-D8 for EDPTL+



3611

NOTE: MISSING SIGNALS CAN BE FOUND IN SGL ANALYSIS

SGL
1 CYCLE
OP CODE 000005

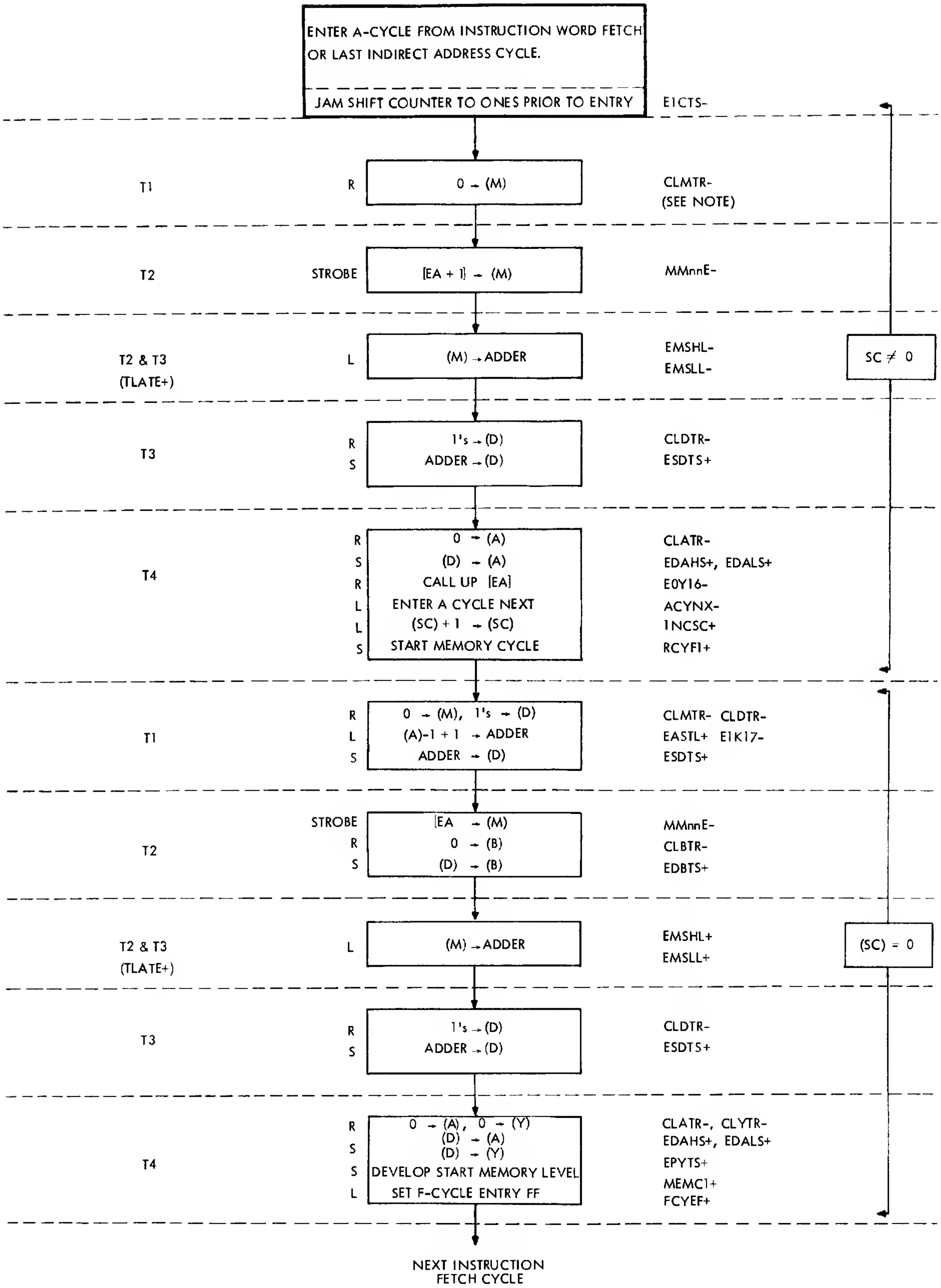
Instruction: Enter Single Precision Mode (SGL)
OP Code: 000005 Type: G, 1 cycle
Description: Reset DPM0D FF

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Execution Time (μsec): 0.96

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
EPSLL+	128-H4	F	$\overline{\text{TLATE}}$	L	(FCYEF+)(TLATE-)	128-F3	101--116-A10	Enable P-register to adder
E1K17-	127-L4	F	$\overline{\text{TLATE}}$	L	(TLATE-)	127-J5	116-D7-D9	Force carry to adder
CLMTR-	128-K8	F	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Clear M-register
CLDTR-	125-J5	F	TL1	R	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	101--116-E7	Clear D-register to ONEs
CLFTL-	125-J7	F	TL1	L	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	120-B1 121-A5 125-L10	Clear F-register Clear shift counter Clear AZZZZ FF
ESDTS+	125-L4	F	TL1	S	(1CYEF-)(ACYEF-)(TL1FF+)	125-A5	101--116-D4-D8	Enable adder sum to D-register
MMnnE-	153/160				(SWnnA+)(STRB1+)	153/160	101--116-H8	Memory data set into M-register
CLPTR-	129-J10	F	TL2	R	(EDPTL+)(MCRST+)*	129-H10	101--116-H10	Clear P-register
EDPTS+	129-L9	F	TL2	S	(EDPTL+)(MCSET+)	129-H9	101--116-G11	Enable D-register into P-register
EASTL+	127-L1	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	101--116-A5	Enable A-register to adder
EMSHL+	127-L8	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	101--108-A8	Enable M(1-7) to adder
ENSHL+	127-L7	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	101--107-A9	Enable M-(1-7) to adder
EMSLL+	127-L10	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	108--116-A8	Enable M(8-16) to adder
ENSLL+	127-L5	F	TLATE	L	(GEN0P+)(TLATE+)(M01FF-)	127-G9	108--116-A9	Enable M-(8-16) to adder
(E1K17+)	127-J6	F	TLATE	L	See gate A1B55-E	127-J6	116-D7-D9 117-B1	Jam carry network
CLDTR-	127-J5	F	TL3	R	(TL3FF+)(ACYLF-)	125-D6	101--116-E7	Clear D-register to ONEs
ESDTS+	125-L4	F	TL3	S	(TL3FF+)(10GRP-)	125-D6	101--116-D4-D8	Enable adder sum to D-register
DPM0D-	124-B10	F	TL3	R	(GEN0B+)(TL3FF+)(M14FF+)(MCRST+)	124-B11	124-B10	Reset DPM0D FF
CLYTR-	129-J3	F	TL4	R	(ACYNX-)(TL4FF+)(MCRST+)	129-D3	101--116-L11	Clear Y-register
EPYTS+	129-L4	F	TL4	S	(P1SEX-)(E01NS+)(0PGJS-)	129-D4/H4	101--116-K11	Enable P-register to Y-register
MEMC1+	126-J11	F	TL4	L	(TL4FF+)(SPM0D-)(TLAFF-)	126-F11/H11	150-C1	Enable set RCYF1+
RCYF1+	150-D1	F	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle

*See gate A1A44-F at 129-D9 for EDPTL+



NOTES: SOME "DON'T CARE" OPERATIONS ARE PERFORMED AT T1 AND T2 WHEN (SC) ≠ 0. THEY ARE OMITTED TO IMPROVE CLARITY.

CPU MUST BE IN DOUBLE PRECISION MODE.

DLD
3 CYCLES
OP CODE 02

Instruction: Double Load (DLD)

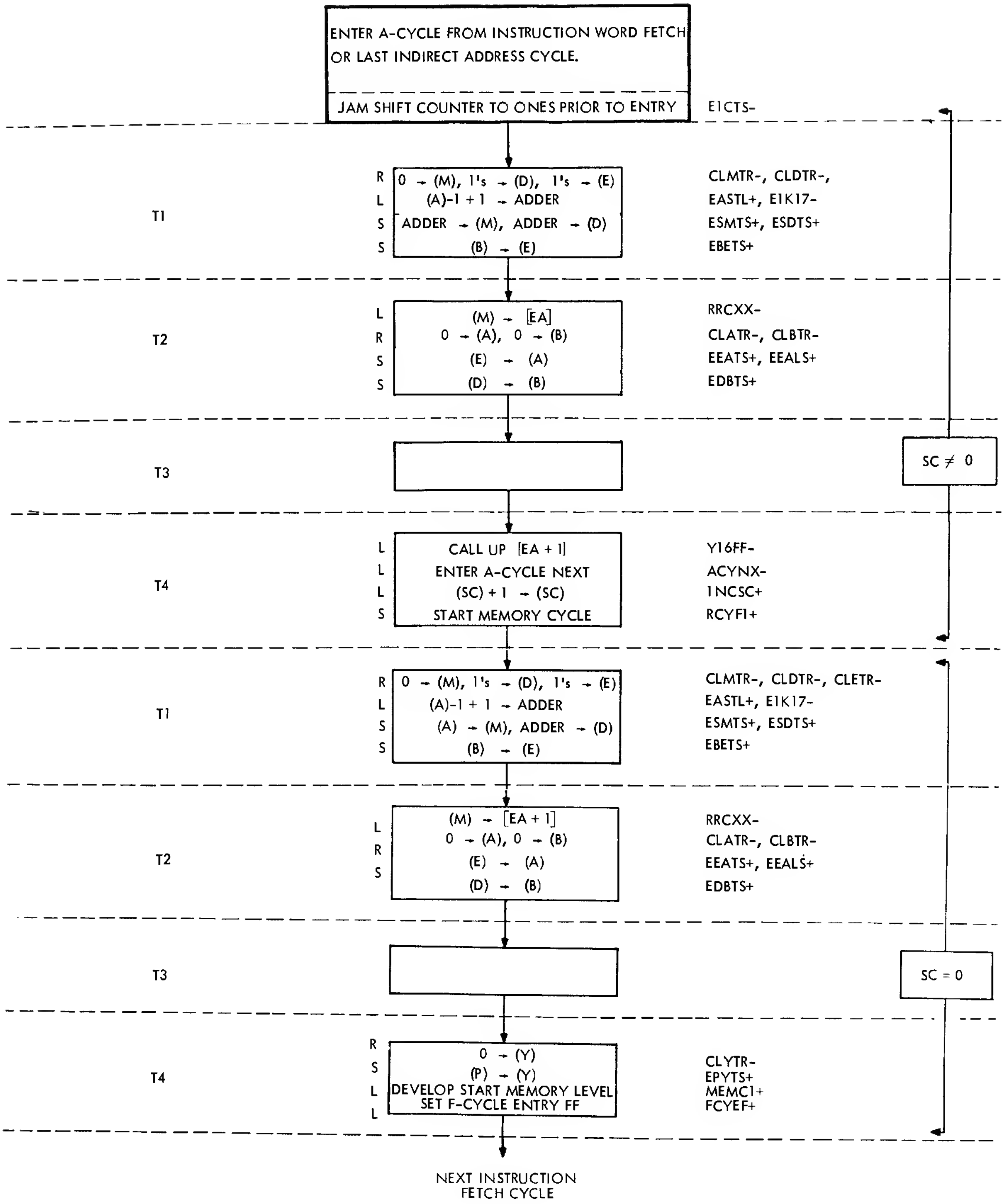
OP Code: 02 Type. MR, 3 cycles

Description: [EA] → (A) (CPU must be in double precision mode)
[EA] + 1 → (B)

F	T	0	0	1	0	S	A	A	A	A	A	A	A	A	A
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Execution Time (μsec): 2.88

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
E1CTS-	125-J8	F	TL4	S	(FCYLF+)(TL4FF+)(0PG3C+)(MCSET+)(AZZZZ-)	125-B7/ H8	121-A9	Set shift counter to 77_8
ACYEF+	119-F4	F	TL4	L	(M01FF+)(TL4FF+)(E01NS-)(F01CY+)	119-F4	119-H3	Set A-cycle at next TL1
CLMTR-	128-K8	A	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Clear M-register
MMnnE-	153/160				(SWnnA+)(STRB1+)	153/160	101--116-H8	Memory data set into M-register
EMSHL+	127-L8	A	TLATE	L	(ACYLF+)(TLATE+)(SUB0P-)(0PGAA+)	127-G8	101--107-A8	Enable M(1-7) to adder
ENSHL+	127-L10	A	TLATE	L	(ACYLF+)(TLATE+)(SUB0P-)(0PGAA+)[(LDA0P-)-]	127-G8	108-116-A8	Enable M-(8-16) to adder
CLDTR-	125-J5	A	TL3	R	(ANA0P-)(TL3FF+)(MCRST+)	125-A6	101--116-F7	Clear D-register to ONEs
ESDTS+	125-L4	A	TL3	S	(TL3FF+)(10GRP-)(MCSET+)	125-D6	101--116-D5	Enable adder sum to D-register
CLATR-	122-J7	A	TL4	R	(ACYLF+)(TL4FF+)(0PGAA+)(1MA0P-)(MCRST+)	122-D2	101--116-H5	Clear A-register
EDAHS+	122-L1	A	TL4	S	(ACYLF+)(TL4FF+)(0PGAA+)(1MA0P-)(MCSET+)	122-D2	101--108-G7	Enable D(1-8) into A(1-8)
EDALS+	122-L2	A	TL4	S	(ACYLF+)(TL4FF+)(0PGAA+)(1MA0P-)(MCSET+)	122-D2	109--116-G7	Enable D(9-16) into A(9-16)
E0Y16-	124-J9	A	TL4	R	(TL4FF+)(MCRST+)(0PGDP+)	124-J9	116-L11	Clear Y-register bit 16
ACYNX-	129-B1	A	TL4	L	(ACYLF+)(LSX0P-)(CAS0P-)(SCZR0-)	129-B1	119-B4	A-cycle next
1NCSC+	126-L3	A	TL4	L	(ACYLF+)(TL4FF+)	126-H4	121-A5	Increment shift counter
MEMC1+	126-J1	A	TL4	L	(TL4FF+)(SPM0D-)(TLAFF-)	126-F11	150-C1	Enable set RCYF1+
RCYF1+	150-D1	A	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle
E1K17-	127-L4	A	$\overline{\text{TLATE}}$	L	(TLATE-)	127-L4	116-D7	Force carry to adder
CLDTR-	125-J5	A	TL1	R	(ACYEF+)(TL1FF+)(JST0P-)(1RS0P-)(1MA0P-)(MCRST+)	125-D4	101--116-F7	Clear D-register to ONEs
EASTL+	127-L1	A	TL1	L	(ACYEF+)(TLATE-)(CAS0P-)(LSX0P-)(10GRP-)	127-J1	101--116-A4	Enable A-register to adder
ESDTS+	125-L4	A	TL1	S	(ACYLF+)(TL1FF+)(JST0P-)(1RS0P-)(1MA0P-)(MCSET+)	125-D4	101--116-D8	Enable adder sum to adder
CLBTR-	123-J6	A	TL2	R	(ACYEF+)(TL2FF+)(DPM0D+)(0PGDP+)(MCRST+)	123-E3	101--116-H2	Clear B-register
EDBTS+	123-L1	A	TL2	S	(ACYEF+)(TL2FF+)(DPM0D+)(0PGDP+)(MCSET+)	123-E3	101--116-G3	Enable D-register into B-register
CLYTR-	129-J3	A	TL4	R	(TL4FF+)(ACYNX-)	129-D3	101--116-L11	Clear Y-register
EPYTS+	129-L5	A	TL4	S	(P1SEX-)(E01NS+)(TL4FF+)(0PGJS-)(MCSET+)	129-D4	101--116-J11	Enable P-register into Ypregister



NOTE: CPU MUST BE IN DOUBLE PRECISION MODE

DST
3 CYCLES
OP CODE 04

Instruction: Double Store (DST)

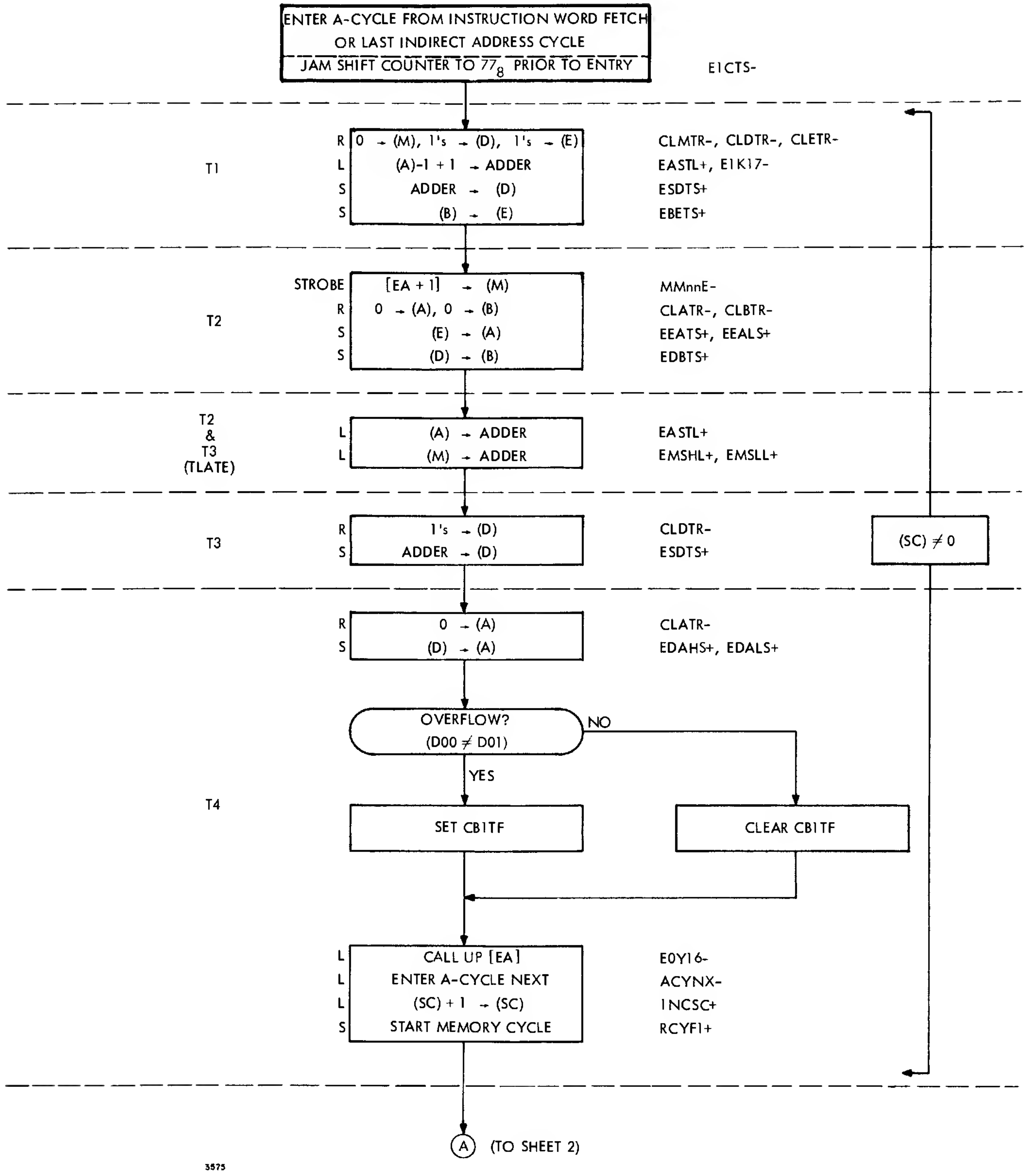
OP Code: 04 Type: MR, 3 cycles

Description: (A) → [EA] (CPU must be in double precision mode)
(B) → [EA] + 1

F	T	0	1	0	0	S	A	A	A	A	A	A	A	A	A
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

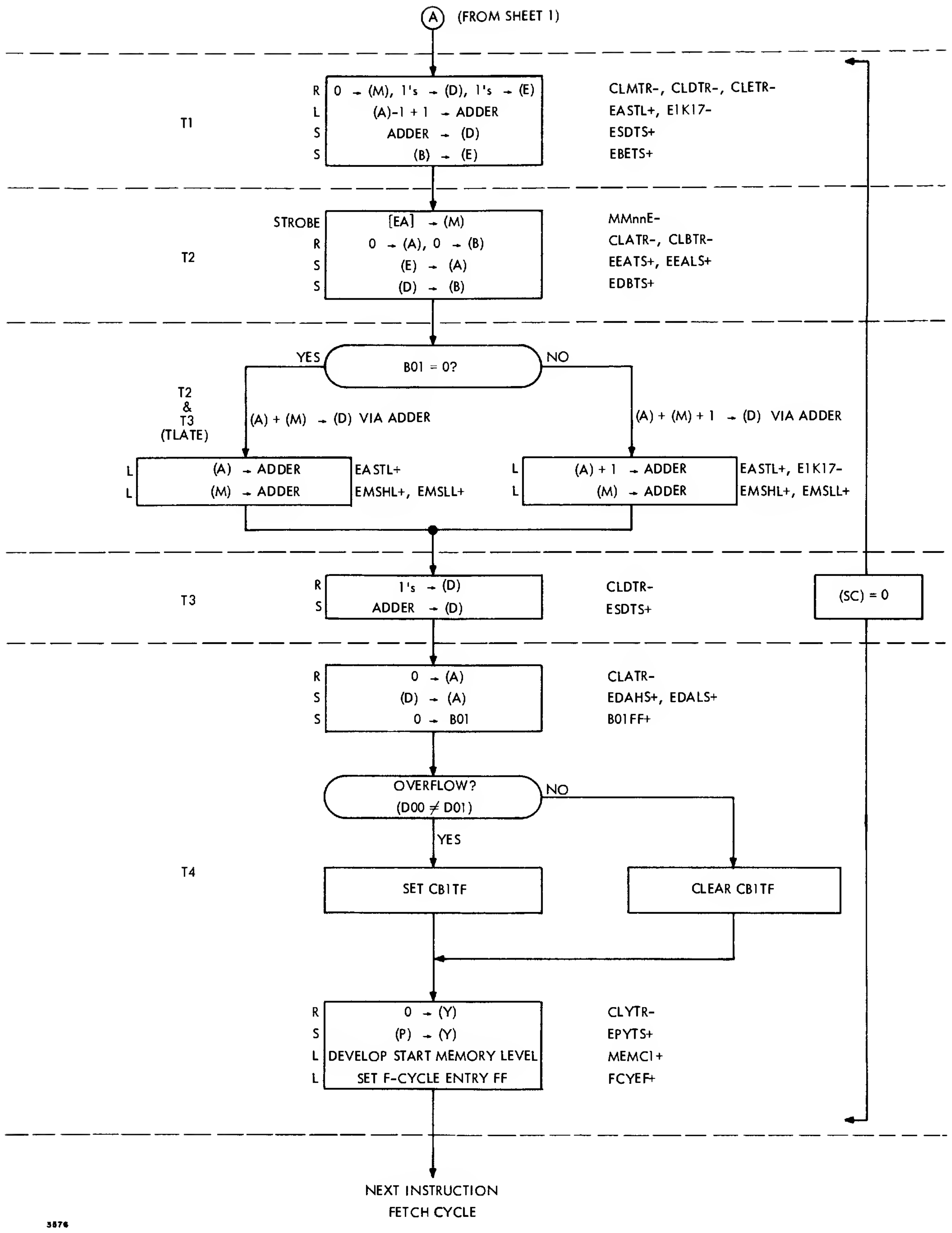
Execution Time (μsec): 2.88

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
E1CTS-	125-J8	F	TL4	S	(FCYLF+)(TL4FF+)(0PG3C+)(MCSET+)(AZZZZ-)	125-B7/H8	121-A8	Set shift counter to 77_8
ACYEF+	119-F4	F	TL4	L	(M01FF+)TL4FF+)(E01NS-)(F01CY+)	119-F4	119-H3	Set A-cycle at next TL1
E1K17-	127-L4	F/A	$\overline{\text{TLATE}}$	L	(TLATE-)	127-J6/J4	116-D7-D9	Force carry to adder
CLMTR-	128-K8	A	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Clear M-register
CLDTR-	125-J5	A	TL1	R	(ACYEF+)(TL1FF+)(JST0P-)(1RS0P-)(1MA0P-)(MCRST+)	125-D4	101--116-F7	Clear D-register to ONEs
CLETR-	125-J2	A	TL1	R	(ACYEF+)(TL1FF+)(DPM0D+)(0PGDP+)(MCRST+)	125-A3	101--116-K3	Clear E-register to ONEs
EASTL+	127-L1	A	TL1	L	(ACYEF+)(TLATE-)(CAS0P-)(LSX0P-)(10GRP-)	127-J1	101--116-A5	Enable A-register to adder
ESDTS+	125-L4	A	TL1	S	(ACYEF+)(TL1FF+)(JST0P-)(1RS0P-)(1MA0P-)(MCSET+)	125-D4	101--116-D7	Enable adder sum to D-register
EBETS+	125-L1	A	TL1	S	(ACYEF+)(TL1FF+)(DPM0D+)(0PGDP+)	125-A3	101--116-J2	Enable B-register to E-register
ESMTS+	128-H10	A	TL1	S	(RRCXX-)(MAST0-)(STA0P-)(TL1FF+)(MCSET+)	128-F10	101--116-G9	Enable A-register into M-register via adder
RRCXX+	126-L6	A		L	(ACYEF+)(0PGWR+)(WR1NH-)	126-F6	150-D6	Block STRB1+ to enable memory write cycle
CLATR-	122-H7	A	TL2	R	(M5G4G+)(MCRST+)	122-H7	101--116-H5	Clear A-register
M5G4G-	123-E2	A	TL2	L	(ACYEF+)(TL2FF+)(DPM0D+)(0PGDP+)	123-E3	122-F4/F7 123-G1/G6	Minterm control for 0PGDP
CLBTR-	123-J6	A	TL2	R	(M5G4G+)(MCRST+)	123-J6	101--116-H2	Clear B-register
EEATS+	122-L3	A	TL2	S	(M5G4G+)(MCSET+)	122-H3	101--110-G4	Enable E(1-10) into A(1-10)
EEALS+	122-K4	A	TL2	S	(EEATS-)	122-K4	111-116-G4	Enable E(11-16) into A(11-16)
EDBTS+	123-L1	A	TL2	S	(M5G4+)(MCSET+)	123-J1	101--116-G3	Enable D-register to B-register
Y16FF-	124-H8	A	TL4	L	(MCSET+)(TL4FF+)(0PGDP+)(0PGSM+)(ACYLF+)(SCZR0-)	124-F9	116-L11	Set Y-register to bit 16
ACYNX-	129-B1	A	TL4	L	(ACYLF+)(LSX0P-)(CAS0P-)(SCZR0-)	129-B1	119-B4	A-cycle next
1NCSC+	126-L3	A	TL4	L	(ACYLF+)(TL4FF+)	126-H4	121-A5	Increment shift counter
MEMC1+	126-J1	A	TL4	L	(TL4FF+)(SPM0D+)(TLAFF-)	126-F11	150-C1	Enable set RCYF1+
RCYF1+	150-D1	A	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle
CLYTR-	129-J3	A	TL4		(TL4FF+)(ACYNX-)	129-D3	101--116-L11	Clear Y-register
EPYTS+	129-L5	A	TL4	S	(P1SEX-)(E01NS+)(TL4FF+)(0PGJS-)(MCSET+)	129-D4	101--116-J11	Enable A-register into Y-register



NOTE: CPU MUST BE IN DOUBLE PRECISION MODE

DAD
3 CYCLES
OP CODE 06
SHEET 1 of 2



DAD
3 CYCLES
OP CODE 06

(SHEET 2 OF 2)

Instruction: Double Add (DAD)

OP Code: 06 Type: MR, 3 Cycles

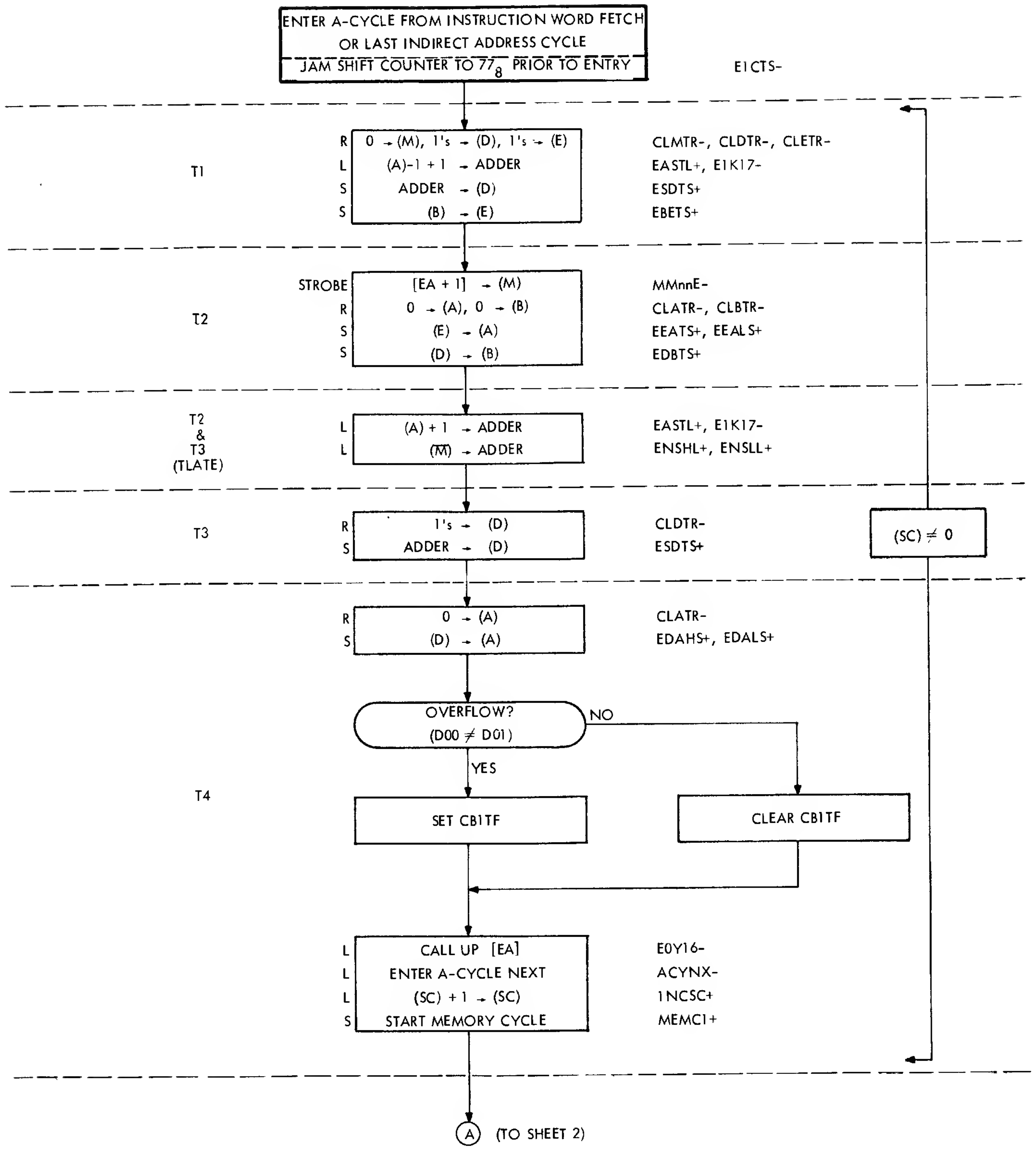
Description: $(A, B) + (EA, EA + 1) \rightarrow (A)_{1-16}, (B)_{2-16}$

$0 \rightarrow B_1$, OVF →(C)

F	T	0	1	1	0	S	A	A	A	A	A	A	A	A	A
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Execution Time (μsec): 0. 96

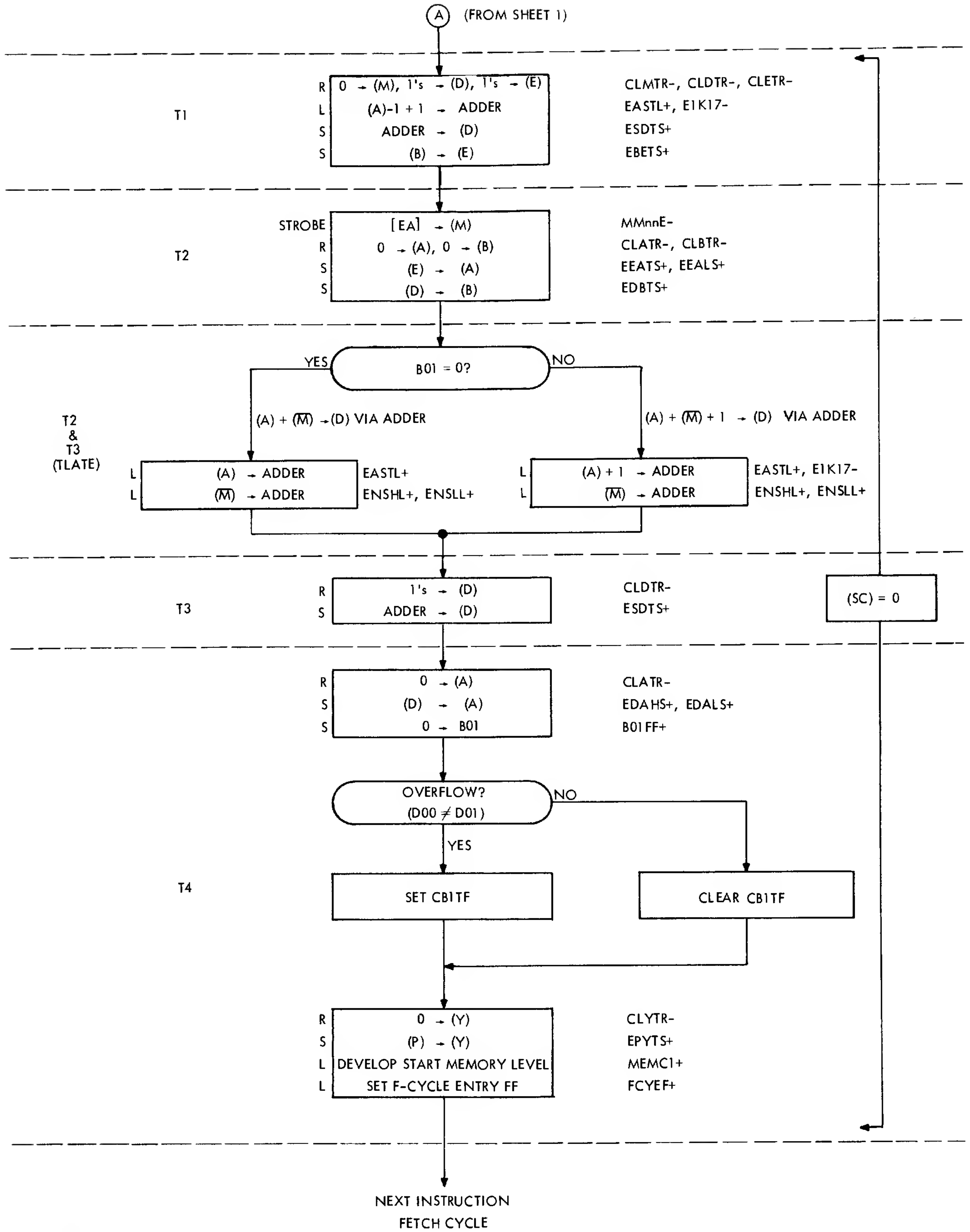
Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
E1CTS-	125-J8	F	TL4	S	(FCYLF+)(TL4FF+)(0PG3C+)(MCSET+)(AZZZZ-)	125-B7/H8	121-A8	Set shift counter to 77_8
ACYEF+	119-F4	F	TL4	L	(M01FF+)(TL4FF+)(E01NS-)(F01CY+)	119-F4	119-H3	Set A-cycle at next TL1
E1K17-	127-L4	F/A	TL1	L	(TLATE-)	127-J6/J4	116-D7-D9	Force carry to adder
CLMTR-	128-K8	A	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Clear M-register
CLDTR-	125-J5	A	TL1	R	(ACYEF+)(TL1FF+)(JST0P-)(1RS0P-)(1MA0P-)(MCRST+)	125-D4	101--116-F7	Clear D-register to ONEs
CLETR-	125-J2	A	TL1	R	(ACYEF+)(TL1FF+)(DPM0D+)(0PGDP+)(MCRST+)	125-A3	101--116-K3	Clear E-register to ONEs
EASTL+	127-L1	A	TL1	L	(ACYEF+)(TLATE-)(CAS0P-)(LSX0P-)(10GRP-)	127-J1	101--116-A5	Enable A-register to adder
ESDTS+	125-L4	A	TL1	S	(ACYEF+)(TL1FF+)(JST0P-)(1RS0P-)(1MA0P-)(MCSET+)	125-D4	101--116-D7	Enable adder sum to D-register
EBETS+	125-L1	A	TL1	S	(ACYEF+)(TL1FF+)(DPM0D+)(0PGDP+)(MCSET+)	125-A3	101--116-J2	Enable B-register to E-register
MMnnE-	153/160				(SWnnA+)(STRB1+)	153/160	101--116-H8	Memory data set into M-register
CLATR-	122-H7	A	TL2	R	(M5G4G+)(MCRST+)	122-H7	101--116-H5	Clear A-register
M5G4G-	123 E2	A	TL2	L	(ACYEF+)(TL2FF+)(DPM0D+)(0PGDP+)	123-E2	122-F4/F7 123-G1/G6	Minterm control for 0PGDP
CLBTR-	123-J6	A	TL2	R	(M5G4G+)(MCRST+)	123-J6	101--116-H2	Clear B-register
EEATS+	122-L3	A	TL2	S	(M5G4G+)(MCSET+)	122-H3	101--110-G4	Enable E(1-10) into A(1-10)
EEALS+	122-K4	A	TL2	S	(EEATS-)	122-K4	111--116-G4	Enable E(11-16) into A(11-16)
EASTL+	127-L1	A	TLATE	L	(ADD0P-)(TLATE+)(ACYLF+)	127-C1	101--116-A4	Enable A-register to adder
EMSHL+	127-L8	A	TLATE	L	(0PGAA+)(SUB0P-)(TLATE+)(ACYLF+)	127-G8	101--107-A8	Enable M(1-7) to adder
EMSLL+	127-L10	A	TLATE	L	(0PGAA+)(SUB0P-)(TLATE+)(ACYLF+)	127-G8	108--116-A8	Enable M(8-16) to adder
E1K17-	127-L4	A	TLATE	L	(DPM0D+)(ADD0P+)(E01NS+)(B01FF+)	127-A4	116-D7/D9 117-B1	Force carry to adder
CLDTR-	125-J5	A	TL3	R	(ANA0P-)(TL3FF+)(MCRST+)	125-A6	101--116-F7	Clear D-register to ONEs
ESDTS+	125-L4	A	TL3	S	(TL3FF+)(10GRP-)(MCSET+)	125-D6	130-F7 101--116-D5-D8	Enable adder sum to D-register
CLATR-	122-J7	A	TL4	R	(ACYLF+)(TL4FF+)(0PGAA+)(1MA0P-)(MCRST+)	122-C3	101--116-H5	Clear A-register
EDAHS+	122-L1	A	TL4	S	(ACYLF+)(TL4FF+)(0PGAA+)(1MA0P-)(MCRST+)	122-C3	101--108-G7	Enable D(1-8) to A(1-8)
EDALS+	122-L2	A	TL4	S	(ACYLF+)(TL4FF+)(-PGAA+)(1MA0P-)(MCSET+)	122-C3	109--116-G7	Enable D(9-16) to A(9-16)
CB1TF+	124-L2	A	TL4	S	(D00FF+)(D01FF+)V(D00FF-)(D01FF-)Λ(ADD0P-)(TL4FF+)(MCSET+)	124-D2/H2	132-C7	Overflow
E0Y16-	124-J9	A	TL4	L	(MCRST+)(TL4FF+)(0PGDP+)	124-J9	116-L11	Reset Y-register bit 16
ACYNX-	129-B1	A	TL4	L	(ACYLF+)(LSX0P-)(CAS0P-)(SCXR0-)	129-B1	119-B4	A-cycle next
1NCSC+	126-L3	A	TL4	L	(ACYLF+)(TL4FF+)	126-H4	121-A5	Increment shift counter
MEMC1+	126-J1	A	TL4	L	(TL4FF+)(SPM0D-)(TLAFF-)	126-F11	150-C1	Enable set RCYF1+
RCYF1+	150-D1	A	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle
B01FF+	124-D10	A	TL4	S	(ADD0P+)(D1V0P-)(ACYLF+)(TL4FF+)(SCZR0+)(DPM0D+)(MCSET+)	124-D10	101-H1	Clear B-register bit 1
CLYTR-	129-J3	A	TL4	R	(TL4FF+)(ACYNX-)	129-D3	101--116-L11	Clear Y-register
EPYTS+	129-L5	A	TL4	S	(P1SEX-)(E01NS+)(TL4FF+)(0PGJS-)(MCSET+)	129-D4	101--116-J11	Enable P-register into Y-register



3573

NOTE: CPU MUST BE IN DOUBLE PRECISION MODE

DSB
3 CYCLES
OP CODE 07
SHEET 1 OF 2



3574

DSB
3 CYCLES
OP CODE 07
SHEET 2 OF 2

(CPU must be in Double Precision Mode)

Instruction: Double Subtract (DSB)

OP Code: 07 Type: MR, 3 Cycles

Description: (A, B) - (EA, EA + 1) → $(A)_{1-16}$, $(B)_{2-16}$
0 → B_1, OVF → (C)

F	T	0	1	1	1	S	A	A	A	A	A	A	A	A	A
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Execution Time (μsec): 0.96

Signal	Origin	Cyc	Tim	Clk	Signal Component	Origin	Destination	Operation Description
E1CTS-	127-J8	F	TL4	S	(FCYLF+)(TL4FF+)(0PG3C+) (MCSET+)(AZZZZ-)	125-B7/ H8	121-A8	Set shift counter to 77_8
ACYEF+	119-F4	F	TL4	L	(M01FF+)(TL4FF+)(E01NS-) (F01CY+)	119-F4	119-H3	Set A-cycle at next TL1
E1K17-	127-L4	F/A	$\overline{\text{TLATE}}$	L	(TLATE-)	127-J6/ J4	116-D7-D9	Force carry to adder
CLMTR-	128-K8	A	TL1	R	(MCRST+)(H0LDM-)(TL1FF+)	128-K8	101--116-H9	Clear M-register
CLDTR-	125-J5	A	TL1	R	(ACYEF+)(TL1FF+)(JST0P-) (1RS0P-)(1MA0P-)(MCRST+)	125-D4	101--116-F7	Clear D-register to ONEs
CLETR-	125-J2	A	TL1	R	(ACYEF+)(TL1FF+)(DPM0D+) (0PGDP+)(MCRST+)	125-A3	101--116-K3	Clear E-register to ONEs
EASTL+	127-L1	A	TL1	L	(ACYEF+)(TLATE-)(CAS0P-) (LSX0P-)(10GRP-)	127-J1	101--116-A5	Enable A-register to adder
ESDTS+	125-L4	A	TL1	S	(ACYEF+)(TL1FF+)(JST0P-) (1RS0P-)(1MA0P-)(MCSET+)	125-D4	101--116-D7	Enable adder sum to D-register
EBETS+	125-L1	A	TL1	S	(ACYEF+)(TL1FF+)(DPM0D+) (0PGDP+)(MCSET+)	125-A3	101--116-J2	Enable B-register to E-register
MMnnE-	153/160				(SWnnA+)(STRB1+)	153/160	101--116-H8	Memory data set into M-register
CLATR-	122-H7	A	TL2	R	(M5G4G+)(MCRST+)	122-H7	101--116-H5	Clear A-register
M5G4G-	123-E2	A	TL2	L	(ACYEF+)(TL2FF+)(DPM0D+) (0PGDP+)	123-E2	122-F4/F7 123-G1/G6	Minterm control for 0PGDP
CLBTR-	123-J6	A	TL2	R	(M5G4G+)(MCRST+)	123-J6	101--116-H2	Clear B-register
EEATS+	122-L3	A	TL2	S	(M5G4G+)(MCSET+)	122-H3	101--110-G4	Enable E(1-10) into A(1-10)
EEALS+	122-K4	A	TL2	S	(EEATS-)	122-K4	111-116-G4	Enable E(11-16) into A(11-16)
EASTL+	127-L1	A	TLATE	L	(SUB0P+)(TLATE+)(ACYLF+)	127-C1	101--116-A4	Enable A-register to adder
ENSHL+	127-L7	A	TLATE	L	(0PGNS+)(1RS0P-)(TLATE+) (ACYLF+)	127-C11	101--107-A9	Enable M-(1-7)to adder
ENSLL+	127-L5	A	TLATE	L	(0PGNS+)(1RS0P-)(TLATE+) (ACYLF+)	127-C11	108--116-A9	Enable M-(8-16) to adder
E1K17-	127-L4	A	TLATE	L	(ACYLF+)(SUB0P-)(JAMKN-)	127-A6/ E7	116-D7/D9 117-B1	Force carry to adder
E1K17+	127-L4	A	TLATE	L	(SCZR0+)(B01FF+)(DPM0D+) (SUB0P+)	127-A6	116-D7/D9 117-B1	No carry to adder
CLDTR-	125-J5	A	TL3	R	(ANA0P-)(TL3FF+)(MCRST+)	125-A6	101--116-F7	Clear D-register to ONEs
ESDTS+	125-L4	A	TL3	S	(TL3FF+)(10GRP-)(MCSET+)	125-D6	130-F7 101--116-D5-D8	Enable adder sum to D-register
CLATR-	122-J7	A	TL4	R	(ACYLF+)(TL4FF+)(-PGAA+) (1MA0P-)(MCRST+)	122-C3	101--116-H5	Clear A-register
EDAHS+	122-L1	A	TL4	S	(ACYLF+)(TL4FF+)(0PGAA+) (1MA0P-)(MCSET+)	122-C3	101--108-G7	Enable D(1-8) to A(1-8)
EDALS+	122-L2	A	TL4	S	(ACYLF+)(TL4FF+)(0PGAA+) (1MA0P-)(MCSET+)	122-C3	109--116-G7	Enable D(9-16)to A(9-16)
CB1TF+	124-L2	A	TL4	S	(D00FF+)(D00FF-)V(D00FF-) (D01FF-)Λ(SUB0P-) (TL4FF+)(MCSET+)	124-D2/ H2	132-C7	Overflow
E0Y16-	124-J9	A	TL4	L	(MCRST+)(TL4FF+)(0PGDP+)	124-J9	116-L11	Reset Y-register bit 16
ACYNX-	129-B1	A	TL4	L	(ACYLF+)(LSX0P-)(CAS0P-) (SCZR0-)	129-B1	119-B4	A-cycle next
INCSC+	126-L3	A	TL4	L	(ACYLF+)(TL4FF+)	126-H4	121-A5	Increment shift counter
MEMC1+	126-J1	A	TL4	L	(TL4FF+)(SPM0D-)(TLAFF-)	126-F11	150-C1	Enable set RCYF1+
RCYF1+	150-D1	A	TL4	S	(MCSET+)(MEMC1+)(RCYF1-)	150-C2	150-D1	Start memory cycle
B01FF+	124-D10	A	TL4	S	(SUB0P+)(D1V0P-)(ACYLF+) (TL4FF+)(SCZR0+)(DPM0D+) (MCSET+)	124-D10	101-H1	Clear B-register bit 1
CLYTR-	129-J3	A	TL4	R	(TL4FF+)(ACYNX-)	129-D3	101--116-L11	Clear Y-register
EPYTS+	129-L5	A	TL4	S	(P1SEX-)(E01NS+)(TL4FF+) (0PGJS-)(MCSET+)	129-D4	101--116-J11	Enable P-register into Y-register